TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Řízení mobilního robota pro vnitropodnikovou logistiku

Diplomová práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Autor práce: **Bc. Jan Brzobohatý**
Vedoucí práce: doc. Ing. Josef Černohorský, Ph.D.

Liberec 2017

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Interoperational logistics mobile robot control

Master thesis

Study programme: N2612 – Electrical Engineering and Informatics
Study branch: 1802T007 – Information Technology

Author: **Bc. Jan Brzobohatý**
Supervisor: doc. Ing. Josef Černohorský, Ph.D.

Liberec 2017

Technická univerzita v Liberci
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2016/2017**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Jan Brzobohatý**
Osobní číslo:	**M15000159**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Informační technologie**
Název tématu:	**Řízení mobilního robota pro vnitropodnikovou logistiku**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Vypracujte rešerši lokalizačních systémů vzhledem ke specifickým požadavkům vnitropodnikové logistiky.
2. Navrhněte software pro řízení několika servisních mobilních robotů.
3. Software realizujte a ověřte na modelu mobilního servisního robota.

Rozsah grafických prací:	**dle potřeby dokumentace**
Rozsah pracovní zprávy:	**40–50 stran**
Forma zpracování diplomové práce:	**tištěná/elektronická**

Seznam odborné literatury:

[1] **Křivý I., Kindler E.: Simulace a modelování. Ostravská univerzita 2001.**

[2] **Hrúz B., Zhou M.: Modeling and Control of Discrete-event Dynamic Systems, Spinger 2007, ISBN 978-1-84628-877-7**

Vedoucí diplomové práce:	**doc. Ing. Josef Černohorský, Ph.D.** Ústav mechatroniky a technické informatiky
Konzultant diplomové práce:	**Ing. Petr Školník, Ph.D.** Ústav mechatroniky a technické informatiky
Datum zadání diplomové práce:	**10. října 2016**
Termín odevzdání diplomové práce:	**15. května 2017**

Technická univerzita v Liberci

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2016

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Poděkování

Děkuji všem, bez kterých by tato práce nemohla vzniknout. Především děkuji panu doc. J. Černohorskému za vedení této práce a cenné rady při jejím vypracovávání. Dále bych chtěl poděkovat Mgr. Nině Adlerové za textovou korekturu.

Abstrakt

Hlavním cílem diplomové práce je navržení a realizace celkového systému pro řízení mobilních robotů pro vnitropodnikovou logistiku. Přičemž důraz je v tomto řešení kladen na nízkou cenu a zejména flexibilitu systému. Byla provedena rešerše lokalizačních systémů, na jejímž základě byl vybrán lokalizační systém využívající měření doby přenosu ultrazvukového signálu, který nejlépe vyhovuje zadaným kritériím. Byl navržen a implementován kompletní systém sestávající z lokalizačního systému, centrální aplikace ovládající celý systém a programu pro ovládání akčních prvků konkrétního robotického zařízení. Celý systém byl úspěšně implementován a otestován na jednom modelu robotického zařízení.

Klíčová slova: robot, logistika, ovládání, řízení, mikropočítač, lokalizační systém, ultrazvuk

Abstract

The main aim of this master thesis is to design and create an overall mobile robot control system for industry logistic. The emphasis in this solution is placed on a low price and especially on the flexibility of the system. A localization system search was made and then was selected the one using ultrasound signal transmission time measuring which suited the specified criteria the best. The complete system consisting of the localization system, the central application controlling the entire system and the program for controlling the action elements of a particular robotic device has been designed and implemented. The whole system has been successfully implemented and tested on one robotic model.

Key words: robot, logistics, control, microcomputer, localization system, ultrasound

Obsah

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

C#	programovací jazyk C Sharp
PWM	Pulse Width Modulation (Pulsně šířková modulace)
RAM	Random-access memory (Paměť s náhodným přístupem)
DDR	Double data rate (Dvojitý datový tok)
AP	Access point (Přístupový bod)
TCP/IP	Transmission Control Protocol/Internet Protocol
REST	Representational State Transfer
API	Application Programming Interface (Aplikační programové rozhraní)
HTTP	Hypertext Transfer Protocol (Hypertextový přenosový protokol)
UDP	User Datagram Protocol
GUI	Graphical User Interface (Grafické uživatelské rozhraní)
XML	Extensible Markup Language (rozšiřitelný značkovací jazyk)

Úvod

Obsahem této diplomové práce je navržení a implementace vnitropodnikového logistického systému. Účelem kompletního systému má být pohyb více robotických vozíků v prostorách výrobní haly bez jakéhokoli lidského zásahu. Instalace samotného systému by měla být co nejméně finančně náročná a zejména by neměla vyžadovat velký zásah do vybavení haly. Dalším důležitým aspektem systému by měla být jednoduchá změna tras a dalších nastavení systému.

V první části této práce je vypracována rešerše lokalizačních systémů. Jsou zde rozebrány absolutní i relativní lokalizační systémy s důrazem na analýzu kritérií, která jsou pro tuto práci klíčová. Je tedy kladen důraz na cenu celého lokalizačního systému a zejména na jeho flexibilitu. Dále je podrobně popsána funkce již konkrétního lokalizačního systému, který byl vybrán jako vhodný kandidát pro realizaci praktické části této práce. Vzhledem k tomu, že lokalizační systém poskytuje informaci pouze o poloze robota v prostoru, ale neposkytuje informaci o jeho orientaci, je zde vypracována i rešerše technologií a metod pro detekci orientace. Na základě této rešerše je opět vybrán vhodný kandidát pro detekci orientace v této implementaci.

Při známé technologii lokalizace je možné navrhnout architekturu celého systému od centrálního řízení celé logistiky až po řízení a pohyb konkrétního robota. V další části práce je tedy rozebrán celý koncept systému, kde je definováno na obecné úrovni, jaké softwarové a hardwarové prostředky budou použity a zejména jakým způsobem spolu budou jednotlivé prostředky komunikovat.

Po navržení celého konceptu systému je potřeba vybrat konkrétní hardwarové prostředky. V další části práce jsou tedy popsány konkrétní řídící prvky a periferie, které byly využity, a je zde zdůvodněn i jejich výběr. Následně je v této práci popsáno, jak jsou jednotlivé prvky propojeny, jakými protokoly spolu vzájemně komunikují i jak jsou jednotlivé prvky napájeny.

Klíčovou částí této diplomové práce je vytvoření aplikace, která bude celý systém ovládat, a logiky pro ovládání jednotlivých robotů. V práci je tedy věnována celá kapitola popisu funkce centrální aplikace s názvem Factory Sheduler a zároveň je zde rozebrána i její architektura. Kromě této centrální aplikace byla ještě pro simulační účely vytvořena aplikace, která simuluje speciální body na mapě, aby bylo možné aplikaci Factory Sheduler testovat bez napojení na reálný systém. A kromě těchto dvou desktopových aplikací je zde podrobně popsána logika, která běží již přímo na mikropočítači, který je umístěn na každém robotickém zařízení. Klíčovou součástí programu na mikropočítači je zejména pohybový algoritmus, který je zde rozebrán detailně.

V následující části je rozebrána cenová náročnost celého systému i jednotlivých robotických zařízení. V poslední kapitole jsou uvedeny veškeré technické překážky a problémy, které musely být v rámci této práce překonány a vyřešeny.

1 Lokalizace

Pro řízení autonomního robotického zařízení je zapotřebí znát zejména jeho aktuální polohu v rámci nějakého prostoru. Přesná lokalizace je jeden z hlavních problémů autonomních systémů, a právě v nich se systémy liší nejvíce. Proto je níže provedena rešerše několika lokalizačních systémů používaných pro autonomní robotické systémy. Pro obecné příkazy z centrálního počítače bude využíváno informací z absolutního globálního lokalizačního systému. Pro konkrétní vykonání pohybu bude každý robot dále využívat relativních lokalizačních metod pomocí informací ze senzorů.

1.1 Rešerše absolutní lokalizace

„Prostředky absolutní lokalizace umožňují zjistit nebo odhadnout absolutní polohu robota v prostoru, bez ohledu na události a stavy, které dosažení této polohy předcházely (Skalka 2011, str. 5).“

Lokalizace pomocí orientačních bodů využívá detekce speciálních nebo přirozených předmětů/rysů pomocí senzorů, přičemž robotovi musí být absolutní pozice těchto předmětů v prostoru známá. Metoda se dá velice zpřesnit pomocí takzvané *trilaterace*, kdy jsou k dispozici tři orientační body a robot dokáže pomocí vzdáleností od každého z nich přesně určit svoji polohu (Skalka 2011, str. 63).

1.1.1 Pasivní orientační body

Jedná se o metodu, kde jsou využívány buď přirozené, nebo uměle vytvořené orientační prvky, které nevysílají ani nepřijímají žádný signál. Jediným požadavkem na takové body je jejich způsobilost být detekován robotem a známost jejich absolutní polohy v prostoru. Většinou se k těmto úkonům využívá strojové vidění. Tato metoda je výpočetně velice náročná, trpí větší chybovostí a může být velice snadno narušena vnějšími podmínkami. Výhodou jsou nenákladné až nulové změny v prostředí, kde se robot pohybuje (Skalka 2011, str. 68).

Laserová navigace

Široce využívanou metodou v této kategorii je laserová navigace, kdy jsou v prostoru rozmístěny odrazové plochy, které mají jedinečné souřadnice. Laserovým vysílačem na robotovi je vyzařován signál do okolí a přijímačem je na robotovi detekován odraz od konkrétních odrazových ploch, přičemž je měřen úhel odrazu a vzdálenost. Tato metoda je velice přesná, flexibilní, ale vyžaduje značné nároky na prostory kvůli viditelnosti odrazových ploch (Štěno 2012, str. 14).

Vodící prvky

Mezi tuto metodu patří i tzv. liniové informační prvky, kdy robot sleduje např. černou čáru na podlaze a pohybuje se tak po určitém grafu (Skalka 2011, str. 69). Tato metoda je finančně velice nenáročná, spolehlivá a ani její výpočetní náročnost není velká. Přičemž realizace vodící čáry může být různá: magnetická, indukční, barevná. Nevýhodou je malá flexibilita.

Inerciální (gyroskopická) navigace

„Tato technologie se nazývá inerciální nebo gyro navigace (obr. 5). Je právě často používána v provozech se značným množstvím regálů, které by mohly blokovat laserový signálu. Každý AGV je vybaven gyroskopem bez pohyblivých částí. Toto zařízení snímá velmi malé odchylky směru jízdy AGV. Stejně jako u laserové navigace, tak i zde je cesta soubor souřadnic uložených v paměti každého AGV. V podlaze jsou podél cesty umístěny malé magnety nebo pasivní RF značky přibližně každých 7,5 m. Tyto body jsou v jedné rovině s povrchem podlahy a je jim přiřazena x, y souřadnice. Tato informace je uložena v paměti AGV. Vestavěný gyroskop zaznamenává malé změny směru jízdy, což je porovnáváno s aktuální uloženou trasou. Na základě toho AGV upravuje svůj směr, aby se udrželo na předepsané trase. Značky v podlaze se používají, jako referenční body ke korekci malých chyb, které se nahromadily mezi jednotlivými značkami. AGV většinou sledují cestu po jednotlivých značkách (Štěno 2012, str. 15).“

1.1.2 Aktivní orientační body

„Snadnou lokalizaci za cenu vyšších zřizovacích nákladů umožňují aktivní orientační body, někdy označované také jako majáčky (anglicky *beacons*), které aktivně vysílají signál nesoucí nějakou lokalizační informaci. K odhadu polohy v prostředí vybaveném majáčky postačuje robotovi signál vysílaný majáčkem přijmout a správně vyhodnotit. Alternativní přístup představují majáčky, které naopak přijímají signál vysílaný robotem a tento samy zpracovávají, anebo na něj robotovi jiným signálem odpovídají (Skalka 2011, str. 64).“

Vlastnosti této metody se znatelně mění podle použité technologie přenosu signálu.

Globální satelitní systémy

Globální satelitní systémy mají tu nevýhodu, že je nelze použít uvnitř budov, a i ve volném prostranství trpí značnou nepřesností a nespolehlivostí. Nicméně jejich značná výhoda je dostupnost po celém světě bez nutnosti investovat do majákových prvků (Skalka 2011, str. 65–67).

Ultrazvukové majákové systémy

Pro určování vzdáleností využívající měření zpoždění signálu při jeho průchodu prostředím se v lokálních lokalizačních systémech často používají ultrazvukové vlny, u kterých je toto zpoždění dobře měřitelné díky jejich rychlosti. Ta je závislá na teplotě i vlhkosti vzduchu, při 20 °C je přibližně rovna 343 m/s. Nevýhodou ultrazvukových vln je jejich poměrně krátký dosah a výskyt nežádoucích odrazů (Skalka 2011, str. 67). Tato metoda dosahuje přesnosti v řádu centimetrů.

WiFi lokalizace

Tato metoda se výrazně liší v různých implementacích. Jedna z nejpřesnějších a zároveň nevyžadující žádné speciální úpravy WIFI AP ani WIFI přijímačů se nazývá SpotFi. Metoda vyžaduje minimálně tři AP, přičemž každý musí mít minimálně tři antény. Tři antény umožňují systému přijmout jak přímý signál, tak různé jeho odražené duplikáty a vyhodnotit, který z přijatých signálů je ten přímý. Poté je provedena trilaterace zpoždění jednotlivých paketů ze všech AP, aby byla zjištěna přesná poloha zařízení. Výhoda WIFI navigačního systému je zejména jeho cena, protože nevyžaduje žádné speciální prvky a může fungovat s použitím běžných WIFI zařízení. Při použití dalších algoritmů pro vylepšení detekce signálu je metoda minimálně náchylná k chybám vlivem odrazu nebo velkých překážek. Tato metoda dosahuje přesnosti v řádu decimetrů (Kotaru, Joshi, Bharadia, Katti 2015, str. 269–271).

Prvním základním stavebním kamenem systému SpotFi je algoritmus pro odhad směru, z kterého přicházejí mnohacestné signály. Aby byl algoritmus použitelný, je zapotřebí signál přijímat minimálně třemi anténami, což je standartní výbava lepších WIFI routerů. Přičemž platí, že čím více anténami zařízení disponuje, tím více mnohacestných signálů je schopné vyhodnotit přesněji. Kromě měření samotného směru je pro lokalizaci měřena doba potřebná pro přenos signálu od vysílače k přijímači (Kotaru, Joshi, Bharadia, Katti 2015, str. 270).

V mnoha případech je však vlivem různých rušivých vnějších vlivů detekce přímé cesty označena jako chybná i přesto, že se ve skutečnosti jednalo o přímou cestu. V těchto případech tedy chybí informace a dochází tak k chybovosti celého algoritmu. Proto je druhým klíčovým prvkem celého systému další algoritmus, který na základ pravděpodobnosti předpokládá, že každá cesta je přímá a až následně je vybírána ta nejlepší (Kotaru, Joshi, Bharadia, Katti 2015, str. 270).

Finální algoritmus již pracuje s informacemi ze všech routerů, přičemž vstupem algoritmu je detekovaný směr a síla signálu z každého routeru. Díky síle signálu dokáže algoritmus přiřazovat jednotlivým routerům váhu, která představuje jejich relevantnost. Systém je díky tomuto postupu

velice robustní a dokáže si poradit s mnoha překážkami (Kotaru, Joshi, Bharadia, Katti 2015, str. 270).

Systém by měl být spustitelný s jakýmkoli dostupným AP na trhu, který poskytuje rozhraní pro získání informací o signálu (CSI – *Channel state infromation*) pro každou anténu zvlášť. Algoritmus SpotFi nahraný do AP pouze odesílá informace (CSI, čas měření a MAC adresu) do centrálního počítače. Při spuštění systému se očekává, že proběhne prvotní jednorázové měření umístění AP, aby bylo následně možné určovat reálnou polohu (Kotaru, Joshi, Bharadia, Katti 2015, str. 271).

1.2 Rešerše relativní lokalizace

„Prostředky relativní lokalizace měří nebo odhadují relativní změnu polohy robota, tedy typicky jeho posunutí a rotaci v rovině vůči jeho předcházející poloze (Skalka 2011, str. 5).“

Všechny metody relativní lokalizace jsou založeny na inkrementální změně nějaké vlastnosti. Vzhledem k tomuto faktu se tyto metody používají spíše jako doplňkové lokalizační systému, protože s ujetou vzdáleností nebo uběhnutým časem se jejich měřící chyba neustále akumuluje (Skalka 2011, str. 38).

Odometrie

Odometrie je metoda odhadu změny pozice a orientace na základě údajů o otáčení kol. K měření se využívá rotační enkodér, který převádí rotační pohyb na dále zpracovatelný elektrický signál. Tato metoda je pro svoji přesnost v praxi velice často využívaná jako doplňkový relativní lokalizační systém. Nicméně přesnost metody je silně závislá na rozlišení snímače a zejména na adhezních poměrech během pohybu (Skalka 2011, str. 38–39).

Senzory využívající Dopplerova jevu

„Dopplerovým jevem nazýváme zdánlivou změnu frekvence přijímaného signálu, pohybuje-li se přijímač vůči vysílači nenulovou rychlostí. Z naměřené změny frekvence je tedy možné spočítat relativní rychlost pohybu (Skalka 2011, str. 56).“

Nevýhoda této metody spočívá zejména v jejím výpočtu, který je založen na integrování naměřené rychlosti podle času, a tudíž chyba měření se zkumuluje v průběhu času, i když se robot nepohybuje (Skalka 2011, str. 57).

Snímkování povrchu

„Zdánlivě rovný povrch pod robotem nebývá typicky úplně hladký, nýbrž zvrásněný drobnými nerovnostmi. Takový povrch lze snímkovat pomocí optoelektronického senzoru a porovnáním po sobě jdoucích snímků zjistit posunutí senzoru po povrchu.“ (Skalka 2011, str. 56–57)

Tato metoda je velice spolehlivá, ale má tři silné nevýhody. Zaprvé nedokáže měřit vyšší rychlost než 6 km/h a zadruhé povrch pro použití této metody nesmí být dokonale hladký (sklo, zrcadlo, lakovaný povrch), ale ani příliš nerovný. Třetí nevýhodou je nemožnost měřit otočení (Skalka 2011, str. 57).

Inerciální navigační systémy

„Inerciální navigace je metoda relativní lokalizace pracující na základě naměřeného zrychlení, z něhož lze integrací vypočítat okamžitou rychlost a z ní pak dále aktuální polohu.“ (Skalka 2011, str. 57)

Pro měření těchto veličin se využívá akcelerometrů a gyroskopů. Výhodou této metody je absolutní nezávislost na vnějším prostředí. Pro správné fungování metody je zapotřebí tříosý gyroskop a tříosý akcelerometr. Kromě toho, že se akumuluje integrační chyba výpočtu, tak do výpočtu vstupují ještě dvě rušivé veličiny. U akcelerometrů se musí u osy z počítat i s gravitační silou, která bude způsobovat, že v ose z nebude nikdy akcelerace nulová. Druhé obtížněji odstranitelnou chybou je otáčení zeměkoule, s kterým je potřeba počítat při vyčítání hodnot z gyroskopu. Zpravidla tato metoda není samostatně použitelná (Skalka 2011, str. 60–61).

Tabulka 1: Porovnání absolutních lokalizačních systémů

Lokalizační systém		Flexibilita	Cenová náročnost		Rušení překážkami	Rušení stroji	Přesnost[1]
			úprav prostoru	jednoho robota			
Absolutní	Laserová navigace	střední	malá	velká	velké	malé	velká
	Vodící prvky	malá	malá	malá	malé	malé	velká
	Inerciální gyrosko-pická navigace	střední	malá	malá	malé	malé	malá
	Ultrazvukové majákové systémy	velká	malá	malá	střední	velké	střední
	Wifi lokalizace	velká	malá	malá	malé	malé	malá
Relativní	Odometrie	velká	malá	malá	malé	malé	malá
	Dopplerův jev	velká	malá	malá	malé	střední	malá
	Snímkování povrchu	velká	malá	střední	malá	malá	malá
	Inerciální navigační systémy	velká	malá	malá	malá	malá	malá

1.3 Marvelmind lokalizační systém

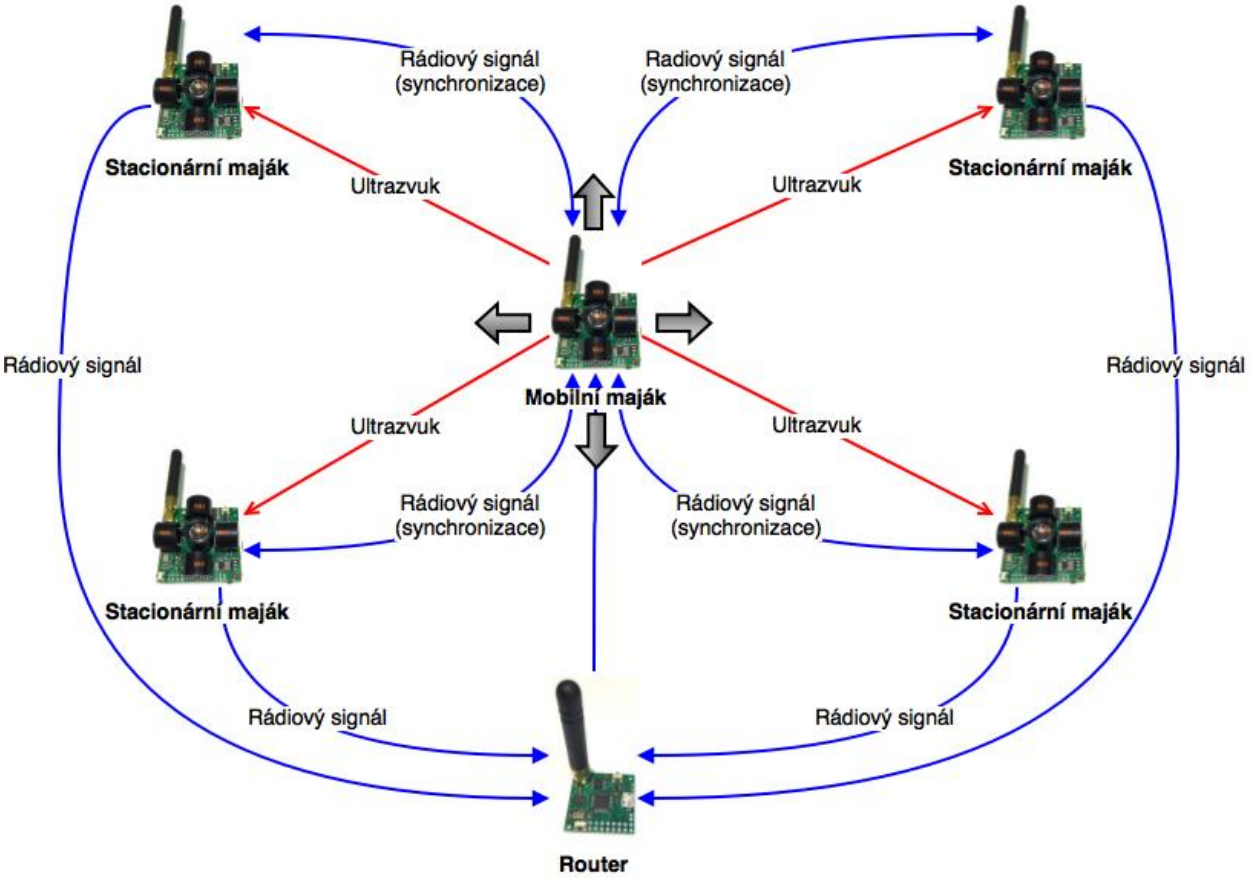
K lokalizaci robotů v rámci výrobní haly byla vybrána metoda ultrazvukových majáků. Primárně proto, že poskytuje dostačující přesnost a nebude pravděpodobně potřeba využívat další lokalizační prvky. Byla zvolena také z toho důvodu, že ve výrobní hale nebude potřeba měnit nijak infrastrukturu majáků, pokud dojde ke změně tras nebo budou-li přidány nějaké překážky. Zároveň se jedná stále o cenově velice dobře dostupnou metodu.

Konkrétně byl vybrán lokalizační systém od firmy Marvelmind robotics, který dosahuje velice dobré přesnosti ±2 cm. Systém se skládá minimálně z pěti ultrazvukových majáků a jednoho routeru. Všechna zařízení jsou vybavena rádiovým modulem s anténou, pomocí nějž spolu komunikují na frekvenci 433 MHz. Čtyři majáky jsou použity jako stacionární a jsou umístěny na

[1] Přesností je zde myšlena dlouhodobá přesnost. Pokud tedy systém trpí akumulační chybou, bude v tomto případě považován za nepřesný i přesto, že v krátkodobém horizonte se může jednat o celkem přesnou metodu.

stěnách nebo jiných předmětech v hale (dále již jen jako stacionární maják). Jeden maják je umístěn na robotickém zařízen, jehož poloha má být lokalizována (dále již jen jako mobilní maják). S touto minimální konfigurací je systém schopen pokrýt až 1000 m^2. Nicméně pokrytí je dále možné zvětšovat tzv. buňkovou architekturou (Marvelmind robotics 2016, str. 4–5).

Systém funguje na principu měření doby přenosu ultrazvukového signálu mezi majáky. Každý stacionární maják přijímá ultrazvukový signál od mobilních majáků a na základě předchozí synchronizace přes rádiový kanál detekuje zpoždění přenosu. Přes rádiový signál je doba přenosu odeslána ze stacionárních majáků do routeru, kde je provedena trilaterace a výsledné souřadnice jsou odeslány zpět do mobilního majáku až 16-krát za sekundu.



Obrázek 1: Schéma komunikace v systému Marvelmind

Frekvence měření pozice mobilního robota může být od 0,5 do 24 Hz v závislosti na nastavení, vzdálenosti a počtu majáků. Vzdálenost mezi sousedními stacionárními majáky by neměla být větší než 50 m. Mobilní maják může komunikovat s robotem přes různá rozhraní: UART[2] přes

[2] Asynchronní sériové rozhraní UART (Universal Asynchronous Receiver and Transmitter). Jde o zařízení pro sériovou komunikaci, jehož nerozšířenější implementací je RS232.

USB nebo SPI[3] přes piny na desce. Zatímco router může s počítačem komunikovat pouze přes USB rozhraní. K běhu celého systému je připojení routeru k počítači zapotřebí pouze při prvotní konfiguraci a pak může být počítač odpojen. Pouze při prvotním vytváření mapy stacionární majáky vysílají i přijímají ultrazvukové pulzy a vyhodnocují svojí skutečnou polohu. Všechna zařízení lze napájet přes USB rozhraní, přičemž majáky mají k dispozici i baterii o kapacitě 1000mAh, která by měla udržet zařízení v pohotovostním režimu okolo 97 hodin (Marvelmind robotics 2016, str. 4–11).

K systému je dodávána aplikace s názvem Dashboard, která umožňuje nastavování jednotlivých prvků systému, aktivaci prvků a zejména detekování mapy stacionárních majáků a sledování pohybu mobilního majáku na mapě (viz Obrázek 2). Mapa se vytvoří zcela automaticky po detekování všech stacionárních majáků bez zadávání jakýchkoli parametrů. Kromě těchto běžně využívaných funkcí aplikace je možné podrobně diagnostikovat pomocí Dashboard různé provozní problémy. Je možné měřit vzdálenost mezi stacionárními majáky a kvalitu signálu mezi nimi. Je možné na softwarovém osciloskopu sledovat přesný průběh signálu a diagnostikovat tak rušivé jevy (Marvelmind robotics 2016, str. 13–19).

Obrázek 2: Aplikace Dashboard od firmy Marvelmind robotics[4]

[3] SPI (Serial Peripheral Interface) je sériové periferní rozhraní. Používá se pro komunikaci mezi řídícími mikroprocesory a ostatními integrovanými obvody (EEPROM, A/D převodníky, displeje…). Komunikace je realizována pomocí společné sběrnice. Adresace se provádí pomocí zvláštních vodičů, které při logické nule aktivují příjem a vysílání zvoleného zařízení.

[4] (Marvelmind robotics 2016, str. 15)

Dashboard aplikace poskytuje rozhraní pro ostatní aplikace přes protokol UDP[5]. Nicméně toto rozhraní je velice omezené a poskytuje pouze funkci pro dotázání se aktuální polohy konkrétního mobilního majáku (podle jeho adresy).

Rozhraní s mnohem větším množstvím možností poskytuje přímo samotný router připojený k počítači přes USB port. Umožňuje měnit konfiguraci routeru, číst souřadnice konkrétních majáků, čtení výšky jednotlivých majáků, aktivace jednotlivých zařízení, nastavování adres a zejména čtení stavu konkrétního zařízení, což zahrnuje i aktuální polohu.

Samotný mobilní maják využívá rozhraní, které nečeká na žádné dotazy, ale vysílá neustálý proud paketů, které buď nesou informaci o tom, že je maják deaktivován, nebo nesou informaci o poslední naměřené poloze majáku.

1.4 Rešerše detekce orientace

Pro správnou funkčnost pohybových algoritmů je zapotřebí znát orientaci zařízení. Protože vybraný lokalizační systém neposkytuje informaci o orientaci zařízení, je potřeba detekovat orientaci jiným způsobem nebo technologií. V této kapitoly budou rozebrány různé přístupy k detekci orientace.

1.4.1 Geomagnetické senzory

„Měří směr magnetického pole Země a převádějí ho na dále zpracovatelný elektrický signál. Geomagnetické senzory mohou pracovat na různých fyzikálních principech, dvě zásadní vlastnosti mají ale společné:

- naměřená orientace je vztažená vůči magnetickému pólu Země,
- jsou náchylné ke zkreslení výsledku měření v důsledku rušení magnetického pole, například v blízkosti konstrukcí z feromagnetických kovů nebo elektrických vedení a strojů.“ (Skalka 2011, str. 75)

1.4.2 Akcelerometr

Pomocí dvouosého akcelerometru je možné měřit aktuální směr pohybu. Jedná se o levné zařízení, ale jeho silnou nevýhodou je měření směru **pohybu,** nikoli směru robota. K určení směru je tedy zapotřebí se zařízením pohybovat.

[5] UDP (User Datagram Protocol) je jeden ze sady protokolů internetu. Na rozdíl od protokolu TCP nezaručuje, zda se přenášený datagram neztratí, zda se nezmění pořadí doručených datagramů, nebo zda některý datagram nebude doručen vícekrát.

1.4.3 Gyroskop

Pomocí dvouosého gyroskopu je možné měřit aktuální rychlost otáčení. Z aktuální rychlosti otáčení je možné vypočítat míru relativní rotace. Jedná se o levné zařízení, ale jeho silnou nevýhodou je měření rychlosti otáčení místo absolutního směru. To sebou nese problematiku přepočtu rychlosti na úhel otočení. Přičemž chybovost je v tomto případě nemalá, ale dá se výrazně redukovat velice krátkým intervalem snímání rychlosti z gyroskopu a přesným měřením času mezi jednotlivými vzorky.

1.4.4 Shrnutí

Ani jedno z výše uvedených řešení detekce orientace není zcela ideální pro implementaci v rámci této práce. První způsob měří orientaci vůči pólu Země, což by vyžadovalo netriviální kalibraci systému, aby bylo možné orientaci vůči pólu převádět na orientaci v rámci souřadného systému majáků. Nicméně ještě větší problém je využívání magnetické pole, které může být výrazně narušeno průmyslovými stroji a velkými kovovými konstrukcemi.

Řešení pomocí akcelerometru nebo gyroskopu má stejnou nevýhodu. Nedostáváme absolutní informaci o směru, ale pouze relativní informaci o změně směru. V obou případech je tedy potřeba udržovat informaci o absolutním směru zařízení a neustále ji aktualizovat podle relativní změny. I s velice malou periodou snímání senzoru bude v tomto případě docházet k akumulaci chyby měření.

Pro řešení této úlohy byl vybrán přístup měření orientace pomocí gyroskopu. Tento způsob byl vybrán z toho důvodu, že se předpokládá nasazení v průmyslových podmínkách, a tudíž se očekává silné rušení magnetického pole, což vylučuje použití magnetometru. Byla dána přednost gyroskopu před akcelerometrem, protože výhody a nevýhody akcelerometru a gyroskopu jsou pro tyto účely principiálně stejné a výpočet otočení z naměřených hodnot gyroskopu je jednodušší.

2 Návrh systému

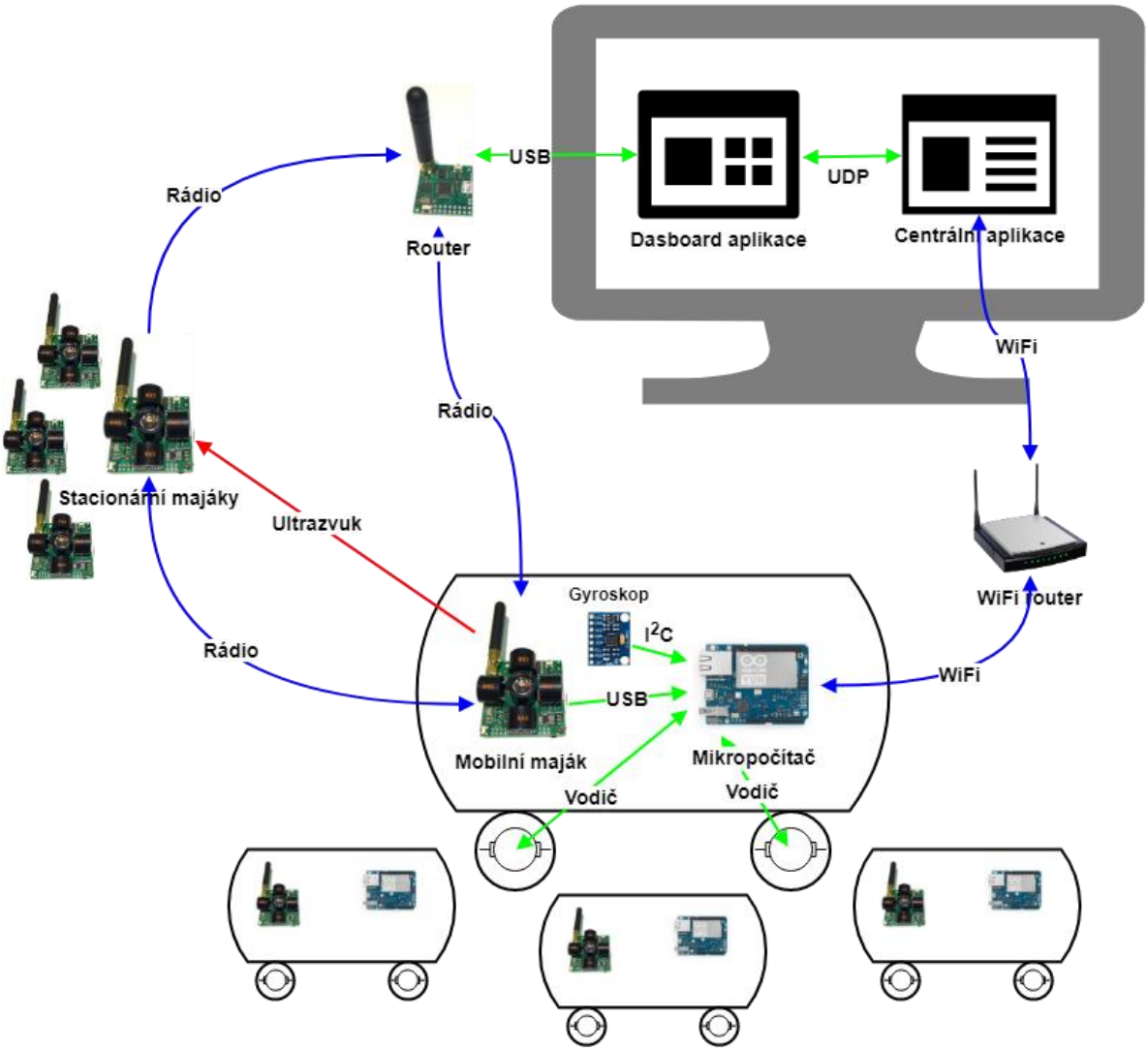
Všechna robotická zařízení budou řízena centralizovaně z hlavního počítače. K přenosu dat mezi počítačem a robotem bude sloužit bezdrátová WIFI síť v prostorách haly. Aby nebyl datový tok přes WIFI síť moc velký a v důsledku toho nedocházelo k častým datovým ztrátám, budou přes síť posílány pouze obecné řídící signály. Respektive bude robotům posílána pouze sekvence bodů, na které se mají aktuálně přesunout v daném pořadí. Co největší část pohybového algoritmu bude řešena na hardwaru umístěném na samotném robotu.

Pro řízení samotného robotického vozíku není zapotřebí velký výpočetní výkon, a proto bylo vybráno v dnešní době moderní, levné a univerzální řešení výpočetní logiky pomocí mikropočítače.

Jak bylo zmíněno výše, komunikace mezi robotem a hlavní aplikací bude probíhat přes WIFI síť. Konkrétně bude probíhat přes protokol TCP/IP. Bude využita architektura REST API[6] typická pro webové služby, která umožňuje publikovat rozhraní přes HTTP protokol. Předpokládá se tedy, že jedno zařízení se bude chovat jako server a druhé jako klient. V tomto systému bude serverem každá jednotka mikropočítače umístěná na robotických vozících a bude přijímat řídící signály nebo odpovídat na dotazy klienta. Přičemž klientem v tomto případě bude centrální řídící počítač.

K lokalizaci robota bude využíván ultrazvukový systém majáků od firmy Marvelmind robotics, jehož výběr je zdůvodněn v kapitole Lokalizace, kde je systém i podrobně popsán. Mobilní ultrazvukový maják umístěný na robotu bude komunikovat přímo s mikropočítačem přes USB rozhraní a bude mu poskytovat aktuální polohu samotného robota. Poloha bude zároveň čtena z routeru centrální aplikací, a to pro každého mobilního robota. Pro zjednodušení bude pro inicializaci celého systému využívána originální aplikace Dashboard (viz kapitola Marvelmind lokalizační systém), která následně dokáže přes UDP rozhraní podávat informace o poloze jednotlivých majáků. Dashboard bude sloužit jako prostředník mezi routerem a centrální aplikací.

[6] „Rozhraní REST je použitelné pro jednotný a snadný přístup ke zdrojům (resources). Zdrojem mohou být data, stejně jako stavy aplikace (pokud je lze popsat konkrétními daty). Všechny zdroje mají vlastní identifikátor URI a REST definuje čtyři základní metody pro přístup k nim. REST implementuje čtyři základní metody, které jsou známé pod označením CRUD, tedy vytvoření dat (Create), získání požadovaných dat (Retrieve), změnu (Update) a smazání (Delete). Tyto metody jsou implementovány pomocí odpovídajících metod HTTP protokolu.“ (Martin Malý 2009)



Obrázek 3: Architektura systému

3 Hardware

Již v první části práce byl popsán systém Marvelmind, který byl použit jako lokalizační systém. V této kapitole budou popsány veškeré zbylé hardwarové prostředky, které v systému byly použity, a bude zde také popsáno jejich vzájemné propojení.

3.1 Mikropočítač

Konkrétně byla v první fázi vybrána platforma Arduino s modelem YUN, jehož podrobný popis a zdůvodnění výběru jsou rozepsány v následující kapitole. Nicméně tato platforma se s postupem času a s postupnou implementací jednotlivých funkcí ukázala jako nevhodná. Mikropočítač neměl dostačující výkon pro vyřešení všech potřebných úloh a zejména nedisponoval dostatečnou programovou pamětí, kterou nebylo možné rozšířit.

Nejprve byly zaznamenány problémy s detekcí stisku tlačítka pro vytváření mapy vlivem nedostatečného výkonu Arduino platformy. Zároveň musel být odstraněn kontrolní součet z knihovny Marvelmind, protože program nebyl schopen zpracovávat vyčítání polohy z mobilního majáku s dostatečnou frekvencí. A to i přesto, že vyčítání polohy majáku byla jediná funkcionalita, která běžela na procesoru, zatímco ostatní úlohy běžely na mikrokontroleru. Následně byl program rozšířen o všechny knihovny a přestal se tudíž vejít do paměti mikropočítače.

Z toho důvodu bylo přistoupeno k mnohem výkonnější platformě Raspberry Pi, jejíž podrobný popis se nachází v kapitole Raspberry Pi.

3.1.1 Arduino

Arduino je open-source platforma s jednoduše použitelným hardwarem a softwarem. Většina Arduino platforem se skládá z malé základní desky, která je osazena procesorem a v závislosti na modelu disponuje různým množstvím a druhem vstupních a výstupních konektorů/zařízení. Primárním cílem je tedy přijímání informací z různých čidel, sensorů, sítí, ovládacích prvků, jejich zpracování a následné publikování dat přes síť nebo ovlivnění vnějších řídících prvků. Arduino také poskytuje vývojové prostředí pro programování algoritmů a jejich nahrávání do zařízení (Arduino S.R.L. 2016a).

Arduino využívá vlastní programovací jazyk, který je procedurální a typový. Lze ho obohatit libovolnými externími knihovnami, které mohou být napsány ve stejném jazyce, nebo případně v jazyce C nebo C++. Jazyk disponuje základními matematickými funkcemi, konstrukcemi pro kontrolu toku programu a mnoha vestavěnými knihovnami zejména pro komunikaci se vstupně

výstupními zařízeními. Programy pro platformu se vyznačují tím, že běží v hlavní nekonečné smyčce (Arduino S.R.L. 2016c).

Arduino YUN

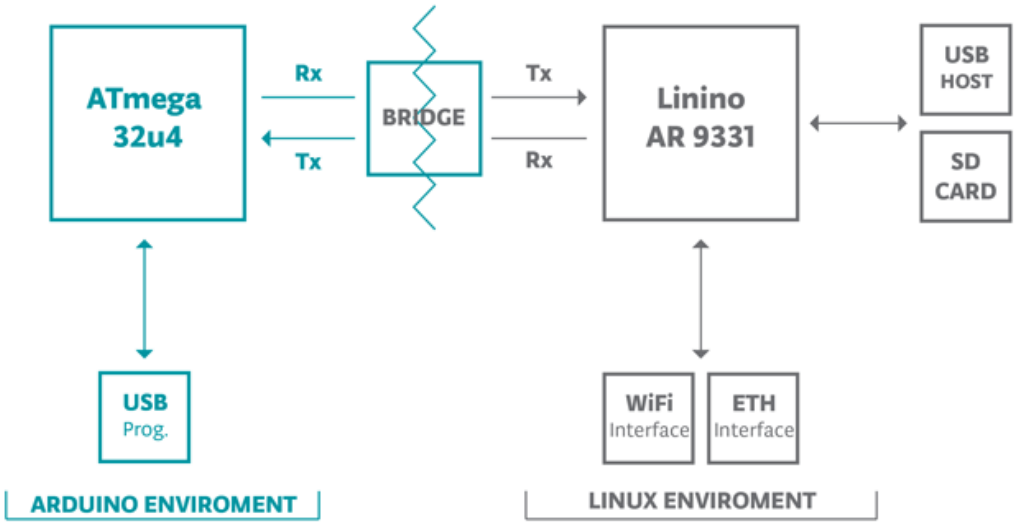
Arduino YUN je jedním z konkrétních modelů platformy Arduino. Model je prodáván s nainstalovanou distribucí Linuxu s názvem Linino OS a právě operačním systémem přímo na desce se nejvíce odlišuje od jiných modelů. Tento model se také vyznačuje především větším počtem vstupně výstupních portů a zařízení, kterými jsou zejména Ethernet, WIFI, USB-A 2.0, USB-micro, micro-SD slot, 20 digitálních vstupně/výstupních pinů (7 může být použito jako PWM výstupy a 12 jako analogové vstupy), 16Mhz oscilátor a 3 reset tlačítka. Jedná se tedy o jeden z nejvíce univerzálních modelů, který dokáže zastat mnoho jednoduchých úkolů (Arduino S.R.L. 2016b).

Obrázek 4: Arduino YUN[7]

Deska vyžaduje napájecí stejnosměrné napětí 5 V (250 mA), které může být připojeno přes USB-micro konektor nebo přes piny na desce. Arduino YUN disponuje 64MB DDR2 RAM a persistentní flash pamětí o velikosti 16 MB, přičemž již z výroby předinstalovaný operační systém zabírá zhruba 9 MB flash paměti. WIFI modul se v továrním nastavení chová jako WIFI AP a je tudíž možné se na něj jednoduše připojit bez dalších zařízení. (Arduino S.R.L. 2016b)

Platforma je osazena jedním procesorem a jedním mikrokontrolerem, přičemž většina vstupně/výstupních zařízení je připojena na procesor (viz Obrázek 5). MIPS procesor pracuje na frekvenci 400 MHz a primárně se stará o chod operačního systému. Mikrokontroler dokáže zpracovávat příkazy přímo v jazyce Arduino, zatímco procesor dokáže využívat různé jazykové interprety. Již z výroby je na linuxovém systému naistalován interpret pro Python 2.7. Vzhledem k tomu, že je většina zařízení připojena přímo na procesor je potřeba při psaní programů v jazyce Arduino přistupovat k těmto zařízením přes můstek (Arduino S.R.L. 2016d).

[7] (Arduino S.R.L. 2016b)



Obrázek 5: Zapojení mikrokontroléru a procesoru na platformě Arduino[8]

Pro plné využití všech výhod modelu Arduino YUN je potřeba si uvědomit, že mikrokontroler běží v tzv. „real-time“ režimu, což je vhodné například k přímému ovládání servomotorů nebo komunikaci s čidly. Zatímco Linux systém na CPU neběží v „real-time“ režimu, ale díky operačnímu systému disponuje mnoha základními funkcemi a ovladači například pro práci s TCP/IP, WIFI připojení (Oberstein 2013).

Platforma Arduino byla vybrána, protože se jedná o platformu, která je velice dobře cenově dostupná a zároveň poskytuje kvalitní vývojové prostředí. Model Arduino YUN byl vybrán z toho důvodu, že je to jeden z mála modelů, který disponuje WIFI modulem. Především však tento model disponuje operačním systémem, na kterém je možné využívat vyšší programovací jazyk pro programování náročnějších aplikací.

3.1.2 Raspberry Pi

Raspberry Pi je mikropočítač se všemi periferiemi integrovanými na jedné desce, přičemž deska má velikost kreditní karty. Oproti Arduino se vyznačuje zejména vyšším výkonem a každý jeho model disponuje místo mikrokontroleru rovnou procesorem. Deska je nejčastěji využívána a také prodávána s předinstalovaným operačním systémem Linux. Speciálně pro Raspberry Pi mikropočítače byla vytvořena distribuce Linuxu s názvem Raspbian, který vychází z distribuce Debian a Pidora vycházející z distribuce Fedora. (Opensource.com 2017).

[8] (Arduino S.R.L. 2016d)

Raspberry Pi je open hardware pouze s výjimkou primárního čipu SoC (Systém on a Chip). Tento čip se stará o veškeré hlavní komponenty mikropočítače jako jsou CPU, grafika, paměť, USB řadič atd. (Opensource.com 2016).

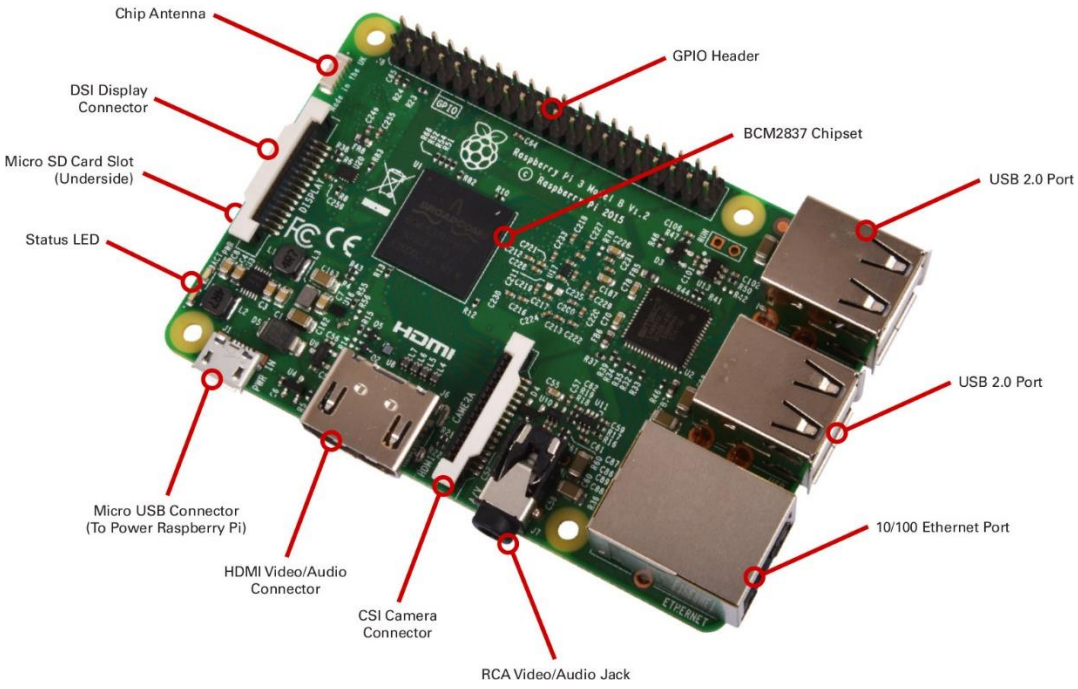
Raspberry Pi 3

Raspberry Pi 3 je nejnovější model Raspberry a disponuje více periferiemi a větším výkonem než předešlé modely, přesto byla zachována zpětná kompatibilita s předchozími modely.

Jednou z hlavních předností je 4-jádrový procesor ARM, který operuje na frekvenci až 1.2 GHz. Aby mohl být plně využit výkon procesoru, je deska osazena DDR2 pamětí RAM s kapacitou 1 GB, která může operovat až na frekvenci 900 MHz. Dále je na desce přítomna jednoduchá grafická karta, která umožňuje používat Raspberry jako plnohodnotný počítač (Chacos 2016).

Co se týče síťového spojení, deska nabízí 100 mbps ethernet konektor, 2.4 GHz WIFI a bluetooth. Kromě toho jsou zde konektory pro připojení monitoru přes HDMI a 3.5 mm jack analogický výstup pro audio/video. Dále disponuje mikropočítač čtyřmi USB 2.0 port (Chacos 2016).

Operační systém je umístěn na microSD kartě, kterou lze libovolně zaměňovat. Instalace systému většinou probíhá pomocí speciálních nástrojů z jiného počítače.

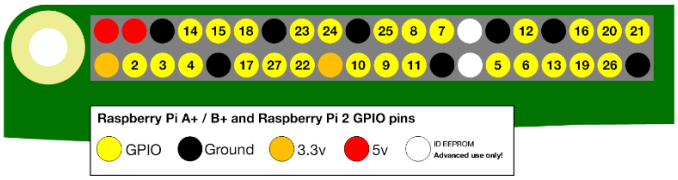


Obrázek 6: Raspberry Pi 3 periferie[9]

[9] (CNXSoft 2016)

Deska je napájena přes mikro USB port, tedy napětím 5 V. Model Raspberry Pi 3 má na úkor svého výkonu jednu výraznou nevýhodu a tou je spotřeba. Doporučuje se napájet desku přímo adaptérem, který je k desce dodán. Tento adaptér dokáže v případě potřeby dodat až 2,5 A proudu, což může být potřeba zejména při zapojení proudově náročných zařízení do USB portů desky.

Deska disponuje také tzv. GPIO (obecně použitelnými vstupy/výstupy). Konkrétně je na desce dostupných 17 takových pinů a dalších 9 pinů pro napájení, přičemž jsou zde piny s 3.3 V i s 5 V napětím. Všechny piny jsou schopné pracovat pouze s digitálním signálem.



Obrázek 7: Raspberry Pi GPIO[10]

3.2 Gyroskop

Jako realizace gyroskopu bylo koupeno čidlo MPU 9250, které je v praxi jedno z nejvyužívanějších univerzálních čidel. Disponuje 3-osím akcelerometrem, 3-osím gyroskopem a 3-osím magnetometrem. Jedná se o velice levné a zároveň ověřené a spolehlivé čidlo. Čidlo lze napájet v rozsahu 3–5 V a lze s ním komunikovat pomocí dvou vodičů přes I²C protokol nebo pomocí 4 vodičů přes SPI protokol. S připojením přes I²C lze dosáhnout rychlosti 400 MHz, ale s SPI lze dosáhnout až 4 KHz (Digi-Key Electronics 2014).

Obrázek 8: Čidlo MPU 9250

Čidlo disponuje 16-bitovými analog/digital převodníky pro digitalizaci 9 os. U gyroskop lze nastavit až 4 přesnosti (±250, ±500, ±1000, ±2000 °/s). Stejně tak u akcelerometru (±2, ±4, ±8, ±16 G). A kompas disponuje jednou pevně danou přesností ±4800 μF. Samotné čidlo detekuje, zda dochází k tzv. prokluzování a případně data samo upravuje na základě korekce dat ze všech devíti os (Digi-Key Electronics 2014).

[10] (Raspberry Pi Foundation 2017)

3.3 RC model

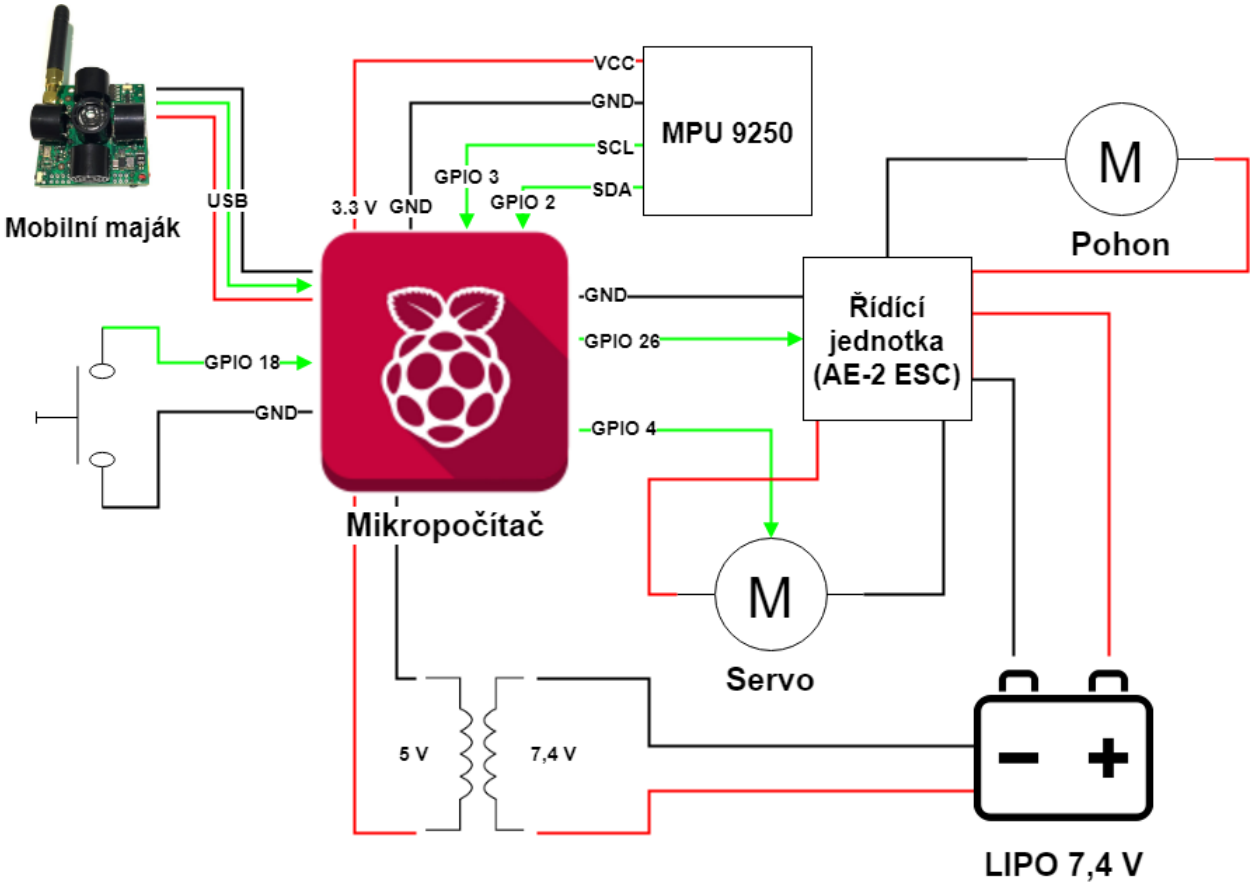
Pro simulaci robotického zařízení byl využit profesionální RC model od firmy Axial. Kostra podvozku modelu je vyrobena z oceli. Auto disponuje přední a zadní nápravou, přičemž jsou obě nápravy napojeny na motor uprostřed kostry pomocí dvou kovových hnacích hřídelí. Součástí pohonné soustavy je také kompaktní převodovka, která může být nastavena od převodu 15:1 až po 74:1 (Axial R/C 2017).

Obrázek 9: RC model

Model je osazen výkonným, přesným a velice citlivým elektrickým motorem 27T opět od značky Axial. O natáčení předních kol se stará citlivý servo motor Tactic TSX45. Oba motory jsou napájeny z řídící jednotky AE-2 ESC, přičemž motor pohonu kol je velikostí tohoto napětí rovnou regulován. Signál z vysílačky je přijímán tříkanálovým přijímačem na frekvenci 2,4 GHz. Do servo motoru je veden řídící signál přímo z přijímače, zatímco signál pro pohon je veden do řídící jednotky, která po demodulaci signálu následně nastaví příslušné napětí na vstupu hnacího motoru. Celá tato elektrická soustava je napájena dvoučlánkovou LiPo baterií s kapacitou 5000 mAh (37 Wh) a napětím 7,4 V (Axial R/C 2017).

3.4 Hardwarové zapojení

Pro funkčnost jednoho robotického zařízení je nutné s mikropočítačem propojit řadu periferií, které poskytují informace o stavu robota, nebo se naopak jedná o akční prvky, které s robotem nějakým způsobem pohybují.



Obrázek 10: Hardwarové zapojení periferií na robotu

Centrální jednotkou je mikropočítač, na který je většina periferií připojena přímo. Výjimkou je pouze motor pohonu kol, který je připojen přes řídící jednotku. Řídící jednotka přijímá signál z mikropočítače v podobě PWM a řídící jednotka tento signál demoduluje a převádí na konkrétní napětí připojené na motor. Podobný postup je u servo motoru pro natáčení předních kol, který přijímá signál také v podobě PWM, ale ten si již signál demoduluje sám, proto je k němu mikropočítač připojen přímo. Nicméně napájen je servo motor také z řídící jednotky.

Celý systém je napájen LIPO dvoučlánkovou baterií s napětím 7,4 V. Přičemž baterie je přímo připojena na řídící jednotku, která se stará o ovládání motorů a zároveň má v sobě napěťový měnič na 5 V pro napájení modelářských servo motorů. Kromě toho je ještě k baterii připojen samotný mikropočítač, který má ale vstup stavěn na napětí 5 V. Tudíž je baterie k mikropočítači připojena přes převodník napětí. Mikropočítač následně rozvádí napětí do dalších periferií jako je například mobilní maják či gyroskopické čidlo.

Čidlo MPU 9250, které disponuje gyroskopem, akcelerometrem a magnetometrem, je k mikropočítači připojeno přes I^2C protokol. Tento protokol vyžaduje propojení přes dva vodiče se signály

SCL a SDA. Přičemž SCL je hodinový signál, pomocí kterého se všechny prvky sběrnice synchronizují, a SDA je datový signál, přes který jsou data přenášena.

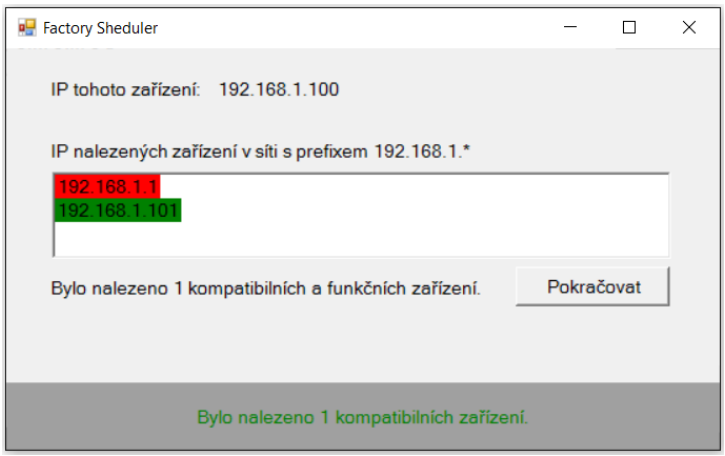
Mobilní maják je připojen přes standardní USB 2.0 port do mikropočítače a přes port USB je také napájen. Tlačítko připojené k mikropočítači slouží k vytváření virtuální mapy, které bude popsáno v dalších kapitolách. Toto tlačítko není povinnou výbavou každého robotického zařízení, ale pouze toho, s kterým se bude vytvářet virtuální mapa. Pro vytváření virtuální mapy dokonce není ani nutné, aby takové zařízení mělo všechny zmíněné periferie. Je zapotřebí pouze k mikropočítači připojit mobilní maják a tlačítko. Nicméně je doporučeno vytvářet mapu s kompletní konfigurací robota, aby byl mobilní maják přesně ve stejné výšce a náklonu při zaznamenávání virtuálních bodů jako při detekci těchto bodů při provozu. Díky tomu je pak možné dosáhnout větší přesnosti.

4 Popis funkce systému

Celý proces začíná spuštěním aplikace Dashboard a detekováním všech stacionárních i mobilních ultrazvukových majáků. Uživatel tzv. zmrazí mapu, čímž se ukončí proces měření vzdáleností mezi statickými majáky a mohou tak přejít do přijímacího módu, kdy je už pouze přijímán ultrazvukový signál z mobilních majáků (viz Obrázek 2).

4.1 Aplikace Factory Sheduler

Po spuštění a nastavení Dashboard aplikace může uživatel spustit centrální aplikaci s názvem Factory Sheduler, která je vytvářena v rámci této diplomové práce a je naprogramována v jazyce C#. Při spuštění začne aplikace automaticky skenovat připojenou WIFI síť. Skenování je provedeno za pomoci diagnostického příkazu *ping*[11], který je aplikován na 254 IP adres, které mají první 3 oktety adresy stejné jako počítač, ze kterého je skenování prováděno. Pokud je na dané IP adrese nějaké zařízení, vrátí příkaz dobu odezvy. Pokud do 12 sekund odezva nepřijde, tak je IP adresa považována za neobsazenou. Všech 254 adres je testováno paralelně, takže celý proces trvá maximálně 12 sekund. Po nalezení obsazených IP adres je na tyto adresy odeslán HTTP dotaz na URL adresu „/check/“ a pokud odpověď na daný dotaz obsahuje řetězec „NVC8mK73kAoXzLAYxFMo“, je zařízení považováno za kompatibilní.



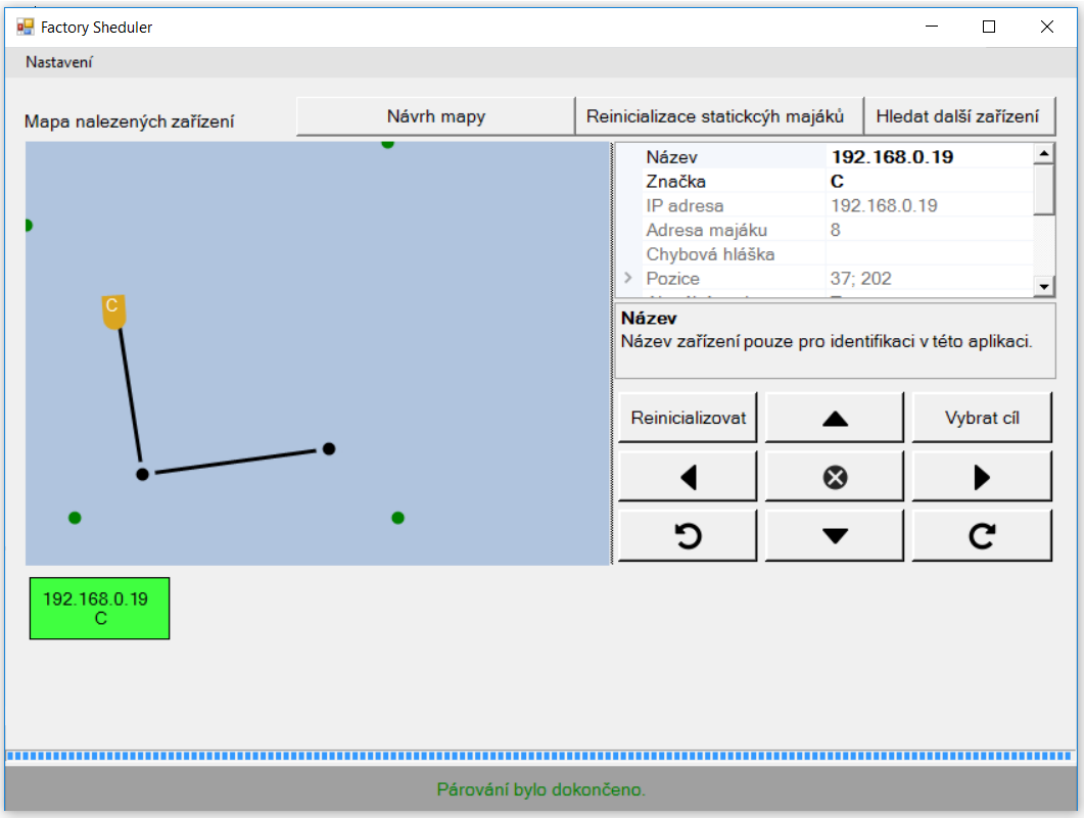
Obrázek 11: Skenování sítě v aplikaci Factory Sheduler

Definice REST aplikačního rozhraní (viz Příloha A) a reakce na toto rozhraní jsou součástí implementace algoritmů na mikropočítači samotného robota (viz kapitola 4.3).

Po nalezení všech kompatibilních zařízení v síti může uživatel přejít pomocí tlačítka „Pokračovat“ na hlavní obrazovku celé aplikace (viz Obrázek 12). Při přechodu na tuto stránku dojde k druhé

[11] Příkaz *ping* s daným argumentem konkrétní IP adresy vrací dobu odezvy zařízení na dané adrese.

fázi inicializace. V dolní části aplikace jsou přidána tlačítka reprezentující jednotlivá zařízení nalezená v předchozím kroku. V první řadě je provedena kontrola připojení k aplikaci Dashboard přes UDP protokol. Pokud není navázáno spojení s Dashboard aplikací, je spuštěna periodická kontrola připojení a v inicializačním procesu se dál nepokračuje, dokud nebude připojení navázáno. Pokud připojení k aplikaci Dashboard proběhlo úspěšně, je při prvním spuštění aplikace uživatel požádán, aby vybral adresy statických majáků z adres všech majáků, které jsou momentálně spuštěny v dosahu routeru. Při dalším spuštění aplikace jsou tyto adresy již načteny z paměti a je možné je dále upravovat v nastavení aplikace. Nicméně ať už adresy byly načteny z paměti nebo je zadal uživatel, dojde ke kontrole připojení vybraných statických majáků a pokud by jeden z nich připojen nebyl, je opět spuštěna periodická kontrola a v procesu inicializace se dál nepokračuje, dokud nebudou všechny připojeny. Závěrečnou částí inicializačního procesu je párování detekovaných robotických zařízení z první fáze inicializace s jejich mobilními majáky. Protože mobilní maják neposkytuje ve svém rozhraní možnost získat jeho adresu, je párování vyřešeno přes porovnávání poloh. Je přečtena hodnota z mobilního majáku pomocí robotického zařízení a přes REST API je odeslána do aplikace. Ve stejnou chvíli je postupně dotazována poloha každého připojeného mobilního majáku přes Dashboard aplikaci. Pokud dojde ke shodě polohy z robotického zařízení a z Dashboard aplikace, je adresa majáku spárována s daným zařízením.



Obrázek 12: Hlavní obrazovka aplikace

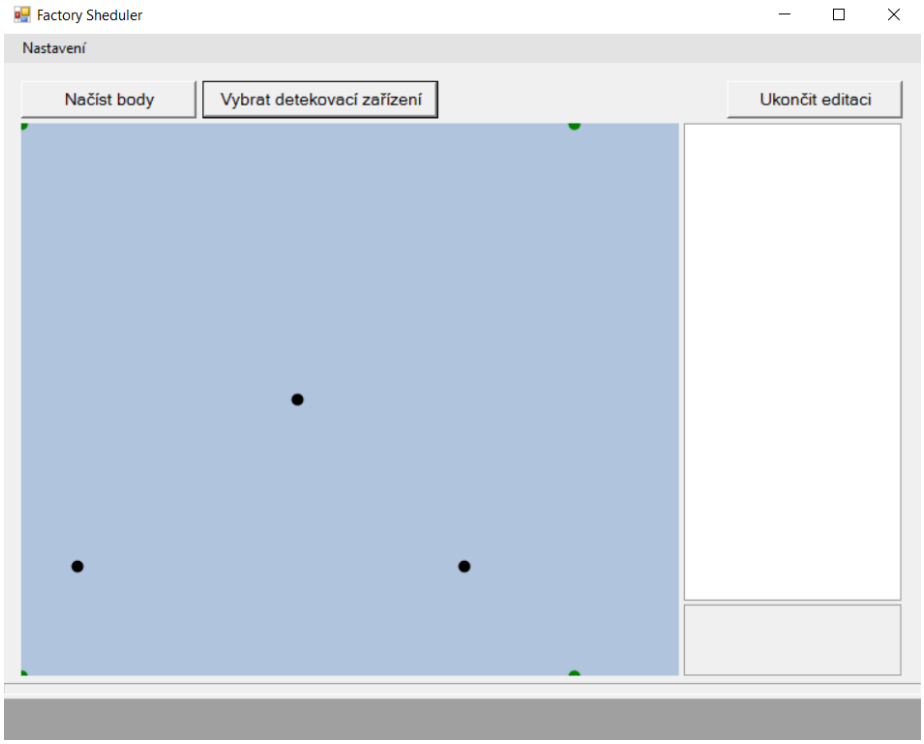
Tím je automatický inicializační proces dokončen. V pravé části obrazovky může uživatel sledovat a měnit aktuální parametry právě vybraného zařízení. Některé parametry je možné měnit a při jejich změně je nová hodnota uložena do EEPROM paměti samotného robota, aby jí bylo možné znovu nahrát při následujícím zapnutí aplikace. V případě jakékoli chyby je chybová hláška vypsána v parametrech konkrétního zařízení a tlačítko zařízení je podbarveno červeně. V opačném případě, tedy je-li vše v pořádku, je tlačítko zařízení podbarveno zelenou barvou.

Zároveň se pod parametry zařízení nachází několik ovládacích tlačítek, která se vztahují také k právě vybranému zařízení. Tlačítkem „Reinicializovat" lze spustit znovu kompletní inicializaci robotického zařízení, která se skládá z detekce API rozhraní a následně spárování s konkrétním mobilním majákem. Tlačítkem „Vybrat cíl" je možné vybrat cílový bod na mapě pro daného robota (viz dále v této kapitole). Dále umožňují 2 šipky dolu a nahoru pohybovat krokově s robotem dopředu a dozadu. Dvě šipky doleva a doprava umožňují relativně natáčet přední kola robota. Tlačítko uprostřed umožňuje zastavit veškerou aktivitu, kterou právě robot provádí. Poslední dvě tlačítka umožňují udělat maximální otočku robota na místě okolo osy z doleva nebo doprava.

Hlavní částí zobrazení je vizualizace prostoru pomocí mapy. Mapa je automaticky po inicializaci nastavena tak, aby pokrývala celý prostor mezi statickými majáky. Statické majáky jsou na mapě zobrazeny v podobě zeleného bodu. Velikosti a pozice elementů na mapě jsou přepočítávány, aby poměrově odpovídaly skutečným rozměrům a pozicím.

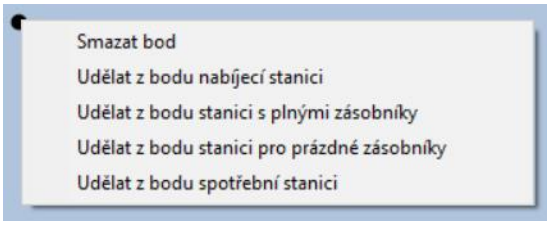
Z přehledu mapy se lze stisknutím tlačítka dostat na obrazovku editace mapy, kde lze vytvářet virtuální cesty, po kterých se budou roboti pohybovat. V tomto kroku musí uživatel nejdříve vybrat, které zařízení[12] bude využito k detekci bodů na mapě. Poté, co uživatel takové zařízení vybere, bude na panelu mapy vizualizována aktuální poloha daného zařízení (žlutá tečka na mapě). Při stisknutí externího tlačítka připojeného na mikropočítač dojde k zaznamenání bodu na mapě do interního bufferu alokovaného v dynamické paměti samotné desky. Pokud uživatel stiskne tlačítko pro detekci bodů, jsou tyto body přes REST API vyčteny z mikropočítače a zobrazeny na mapě (černé tečky).

[12] Zařízením je zde myšlena kombinace mikropočítače s mobilním majákem a stejným softwarem jako disponují roboti, případně může být využita kompletní implementace robota.



Obrázek 13: Editace mapy (detekce bodů)

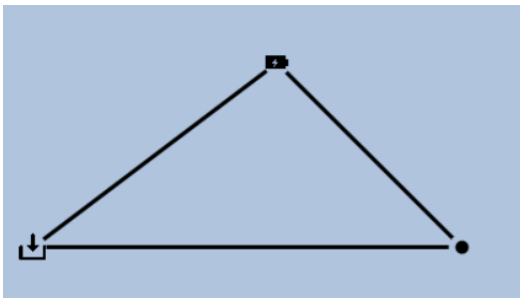
Po načtení bodů je možné upravovat určité vlastnosti bodů, mazat body, vytvářet cesty mezi body nebo případně cesty mazat. Jedním z prvních kroků je změna typu bodu na mapě. Na výběr je z 5 možností (viz Obrázek 14). Výchozí typ (zobrazeno jako černá tečka) bodu je pouze průchozí bod, na který se mohou napojovat cesty, ale nemá žádnou další funkci. Z bodu lze udělat nabíjecí stanici (ikona baterie), což znamená, že robota je možné na daném místě dobít. Další tři typy bodu jsou určeny pro konkrétní práci s obsahem robotického vozíku. Robot může vést buď prázdný zásobník do stanice, kde dochází ke skladování prázdných nádob (ikona čtverce), nebo může nabrat plnou nádobu v naplňovací stanici (ikona se šipkou dovnitř) a dovést ho do stanice, kde dojde k jeho spotřebování (ikona se šipkou ven).



Obrázek 14: Editace mapy (výběr typu bodu na mapě)

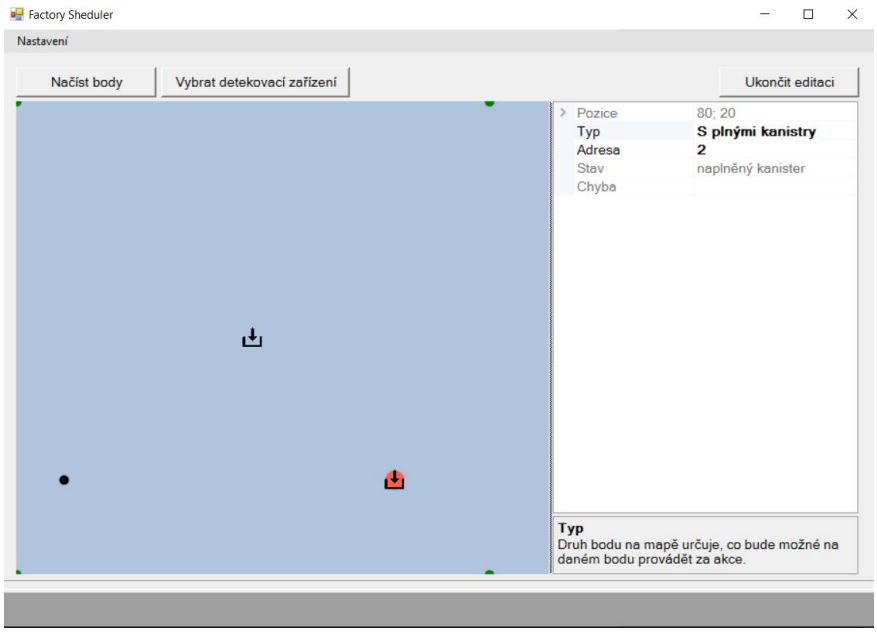
Aby bylo možné určit, kde se lze pohybovat s robotem, musí uživatel jednotlivé body propojit pomocí myši v aplikaci rovnými čarami. Tyto rovné čáry mezi body představují cestu, kterou se může robot vydat. Mezi dvěma body může být maximálně jedna cesta, přičemž směr pohybu po

této cestě je libovolný. Po zadání celé mapy uživatelem vznikne jednoznačný **neorientovaný graf** představující možné cesty pohybu robota.



Obrázek 15: Editace mapy (Zobrazení typů bodů a cest mezi nimi)

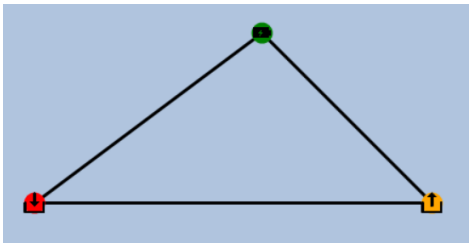
Jednotlivé body lze vybrat přímo kliknutím na mapě a v pravé části obrazovky je následně možné vidět parametry vybraného bodu. Některé ze zobrazených parametrů je možné měnit. Uživatel by měl u speciálních bodů vybrat, jaký typ zařízení se na daném bodu nachází a jakou má virtuální adresu. Virtuální adresa je celočíselná hodnota, která slouží k jednoznačné identifikaci zařízení.



Obrázek 16: Editace mapy (vlastnosti konkrétního bodu)

Aplikace přijímá informace z jednotlivých zařízení přes UDP připojení, přičemž obsahem jednoho požadavku je virtuální adresa zařízení, stav zařízení zakódovaný do jednoho znaku a typ zařízení zakódovaný do jednoho znaku. Aby aplikace takové zařízení zaregistrovala, musí jí zařízení odeslat požadavek. Následně by mělo zařízení při každé změně stavu odeslat nový požadavek aplikaci, aby byla změna zaregistrována. Ovšem aplikace nemůže být závislá pouze na přijatých požadavcích, protože pak by to znamenalo, že po zapnutí aplikace bude muset dojít k odeslání požadavku

z každého zařízení, než aplikace zaregistruje, že takové zařízení vůbec existuje. Proto aplikace po svém spuštění odesílá požadavek na zjištění stavu všech zařízení, která má uložená (viz následující odstavec). Na standartní mapě je následně stav jednotlivých zařízení rozlišován barevně.



Obrázek 17: Indikace stavu speciálních bodů na mapě

Aby nebylo při každém zapnutí aplikace pokaždé potřeba nastavovat celou mapu znovu, dochází k jejímu persistentnímu ukládání do souboru. K uložení dochází ve chvíli, kdy uživatel ukončí editaci mapy. Konkrétně jsou ukládány tři soubory v XML podobě a jedná se o serializované objekty daných prvků. První obsahuje kompletní definici všech bodů na mapě, druhý obsahuje definované cesty mezi těmito body a třetí obsahuje souřadnice statických majáků. Souřadnice statických majáků jsou ukládány pouze pro kontrolu, aby bylo možné při každém spuštění zkontrolovat, zda jsou statické majáky z dashboard stále vyčítány se stejnými souřadnicemi. Pokud by to tak nebylo, je potřeba celou mapu nastavit znovu v novém souřadném systému.

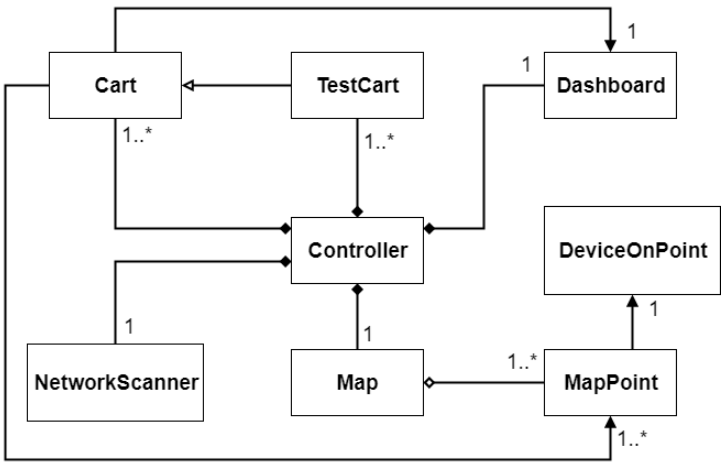
Mapa možných cest prostorem je interpretovaná neorientovaným grafem, kde uzly jsou křižovatky nebo speciální místa, kde se vykonává nějaká činnost, a hrany jsou cesty mezi těmito křižovatkami. Pro hledání nejkratší cesty pro jednotlivé roboty byl vybrán efektivní Dijkstrův algoritmus, jehož složitost je obecně $O(V^2+E)$, kde V je počet uzlů a E je počet hran. Nicméně je možné pro řídké grafy využít jeho mnohem efektivnější podobu, kdy je ukládán graf pomocí seznamu sousedů a pak algoritmus dosahuje složitosti $O(E+VlogV)$. Při vytváření mapy v tomto systému se předpokládá mnohonásobně vyšší počet uzlů než hran, a proto byla využita tato efektivnější podoba Dijkstrova algoritmu.

Ve chvíli, kdy je mapa vytvořena, je možné zadávat konkrétním robotům příkaz, kam se mají pohybovat. To je možné udělat programově nebo pomocí uživatelského rozhraní tlačítkem „Vybrat cíl“. Pokud je tedy vybrán konkrétní robot a je stisknuto tlačítko „Vybrat cíl“, stačí následně na mapě vybrat libovolný bod, kam se má začít robot pohybovat. Algoritmus nejdříve vybere nejbližší bod z mapy k danému robotu a vypočítá nejkratší cestu z tohoto bodu do vybraného cílového bodu na mapě. V tu chvíli je trasa zobrazena na mapě fialově a do robotického zařízení je odeslán

seznam bodů, přes které se má robot postupně pohybovat. O zbytek se již stará algoritmus přímo v mikropočítači robota.

4.1.1 Struktury aplikace

Aplikace Factory Sheduler má značně rozsáhlou strukturu tříd a dalších modulů. Z toho důvodu budou v této kapitole rozebrány pouze klíčové třídy celé aplikace, přičemž jsou úplně vynechány veškeré třídy pro grafické uživatelské rozhraní, které bylo popsáno v předešlé kapitole.



Obrázek 18: Struktura aplikace Factory Sheduler

Hlavní řídící třídou, která řídí celou aplikaci, je třída Controller, která se stará o vytváření většiny hlavních modulů a zároveň zajišťuje jejich předání do jiných tříd. Samotný Controller zajišťuje klíčové algoritmy aplikace, jako je inicializace celého systému, a zpracovává uživatelské příkazy z grafického rozhraní aplikace. K prvotnímu kroku inicializace využívá Controller modul NetworkScanner, který poskytuje veškeré potřebné algoritmy pro skenování WIFI lokální sítě a vyhledávání kompatibilních zařízení.

Controller také drží informaci o mapě pomocí třídy Map, která je interně složena z jednotlivých bodů na mapě a jejich spojení. Pro interpretaci bodů na mapě je využívána třída MapPoint, která umožňuje nastavovat nebo získávat informace o konkrétním bodu. Pokud se navíc jedná o bod, který představuje nějaké operační místo (nabíjecí stanice, plnící stanice atd.), může být vytvořena instance třídy DeviceOnPoint, která slouží k interpretaci tohoto ovládacího místa, zejména tedy ke komunikaci s ním.

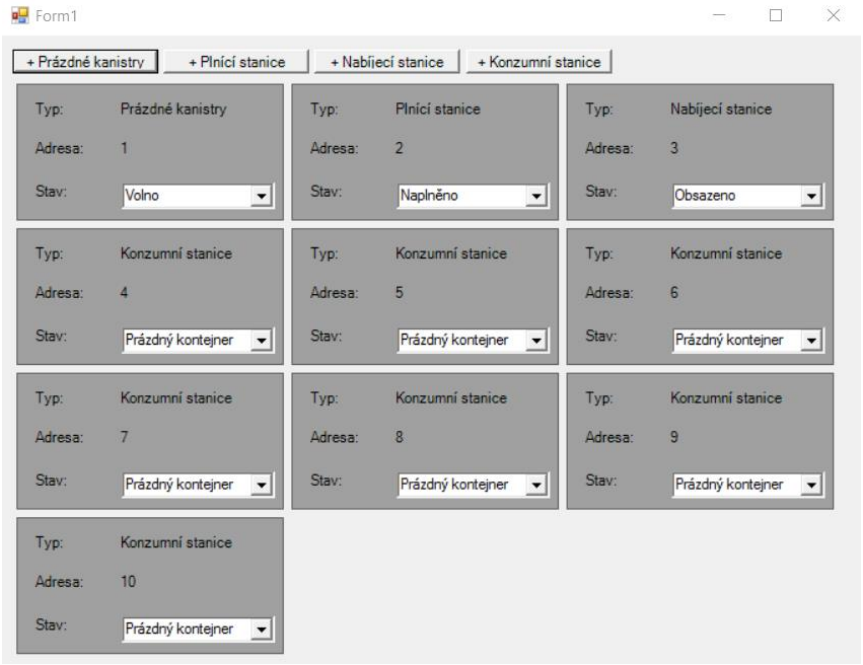
Jedním z nejdůležitějších modulů je třída Cart, která představuje jedno robotické zařízení. Tato třída umožňuje získávat aktuální stav robotického zařízení a zajišťuje veškerou komunikaci s robotem. Pro uchování aktuální cesty pohybu používá seznam bodů na mapě, k čemuž je využita již

zmíněná třída MapPoint. Pro komunikaci s Marvelmind aplikací Dashboard využívá speciální třídu s názvem Dashboard, která veškerou komunikaci s aplikací zajišťuje. Z třídy Cart navíc dědí třída TestCart, která slouží pouze pro testovací účely aplikace. Jedná se o robotická zařízení, která simulují chování skutečného zařízení.

4.2 Simulátor speciálních bodů na mapě

Jak bylo řečeno v předešlé kapitole, aplikace Factory Sheduler přijímá veškeré informace o stavu všech zařízení přes UDP. Indikace těchto zařízení, a tedy i odesílání požadavků do aplikace, není součástí řešení této diplomové práce a nejsou k dispozici ani žádné simulační prostředky těchto zařízení. Proto byla vytvořena další aplikace v jazyce C#, která simuluje UDP požadavky z různých typů zařízení.

Simulační aplikace disponuje jednostránkovým grafickým uživatelským rozhraním, které umožňuje uživateli přidávat různé typy zařízení. Následně je možné u každého zařízení měnit jeho stav. Aplikace automaticky při zapnutí naváže komunikaci přes UDP s aplikací Factory Sheduler.



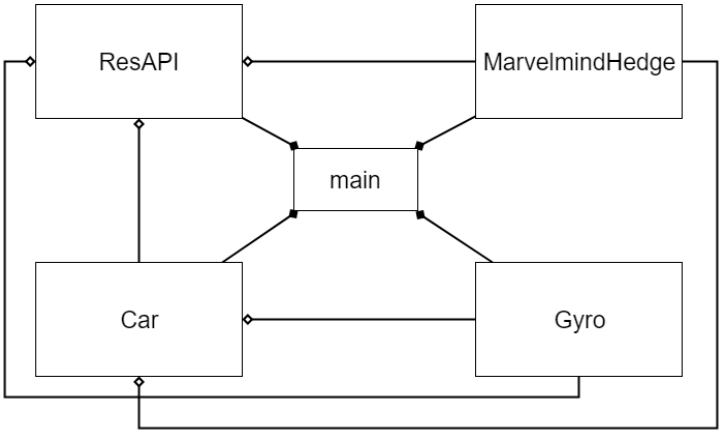
Obrázek 19: Aplikace pro simulaci vstupu do Factory Sheduler

4.3 Mikropočítač

Veškerá logika pro ovládání robota byla naprogramována v programovacím jazyce Python. Tento jazyk byl vybrán, protože vývoj v něm je rychlý a operační systém Raspbian již poskytuje kompletně připravené prostředí pro programování v Pythonu. Jeho nevýhodou je rychlost vykonávání

programu, protože se jedná o plně interpretovaný jazyk, nicméně v tomto případě nebylo očekáváno velké výkonové zatížení pohybovými algoritmy. Výhodou je, že Marvelmind lokalizační systém nabízí již připravenou knihovnu v jazyce Python pro obsluhu mobilního majáku.

Aplikace se skládá ze čtyř hlavních modulů a jednoho spouštěcího modulu. Každý z hlavních 4 modulů je zároveň třídou a každý modul je spuštěn v separátním vlákně. Z každého modulu je v aplikaci vytvořena pouze jedna instance, která je případně sdílena do ostatních modulů, ve kterých je využívána.



Obrázek 20: Vztah modulů na mikropočítači

Spouštěcí „main“ modul se stará o vytvoření všech modulů, jejich spuštění a předání parametrů, což jsou většinou odkazy na ostatní moduly. Kromě vytvoření hlavních modulů se spouštěcí modul stará také o obsluhu hardwarového tlačítka pro vytváření mapy. Pro obsluhu tlačítka je využito metody přerušení, aby se aplikace nezatěžovala cyklickou kontrolou stavu tlačítka. Při každém stisknutí tlačítka (přerušení) je do bufferu zaznamenána aktuální poloha robota. Takto se shromažďují jednotlivé body na mapě, dokud nejsou vyčteny ze zařízení pomocí Rest API.

Rest API modul se stará o správu HTTP serveru, který zpřístupňuje REST rozhraní přes lokální WIFI síť. Tento modul je jediný (tedy kromě spouštěcího modulu), který má odkaz na všechny tři ostatní hlavní moduly. Vyplývá to už z logiky věci, protože skrze rozhraní by mělo být možné ovládat robota i číst jeho stavy. Proto je zapotřebí, aby Rest API modul komunikoval přímo s lokalizačním čidlem, gyroskopickým čidlem a zároveň s ovládáním robota.

Podrobně je REST API pro robota popsáno v příloze A. Rozhraní disponuje funkcí „check“, která vrací vždy stejný kód. Tato funkce slouží pouze pro detekci, že se jedná opravdu o kompatibilní zařízení. Tato funkce je využívána při hledání kompatibilních zařízení v síti. Dále rozhraní poskytuje metody pro detekci stavu „position“ a „heading“. Jak z názvu metod vyplývá, jedná se

o metodu pro získání aktuální pozice robota a o metodu pro získání aktuálního směru robota. Dále jsou zde metody pro přímé ovládání pohybu robota, jako jsou „raid“ pro natočení předních kol, „drive“ pro pohon robota a „turn“ pro otočení robota. Metody pro absolutní ovládání by měly být používány pouze výjimečně a spíše pro testování, protože komunikace přes HTTP je nespolehlivá a pomalá. Pro vyčítání nastřádaných pozic na mapě z bufferu slouží metoda „read-map-points“. Aby nebylo nutné vždy v grafickém rozhraní Factory Sheduleru nastavovat znovu některé parametry zařízení, jako jsou název, alias a vzdálenosti mobilního majáku od okrajů robota, ukládají se tyto uživatelsky nastavené parametry přímo do persistentní paměti robota pomocí metody „writeEEPROM“. Při opětovném spuštění aplikace jsou tyto parametry ze zařízení vyčítány pomocí metody „readEEPROM“. Nakonec disponuje rozhraní nejdůležitější metodou „path“, pomocí které je možné robotovi nastavit, přes které body na mapě se má postupně vydat.

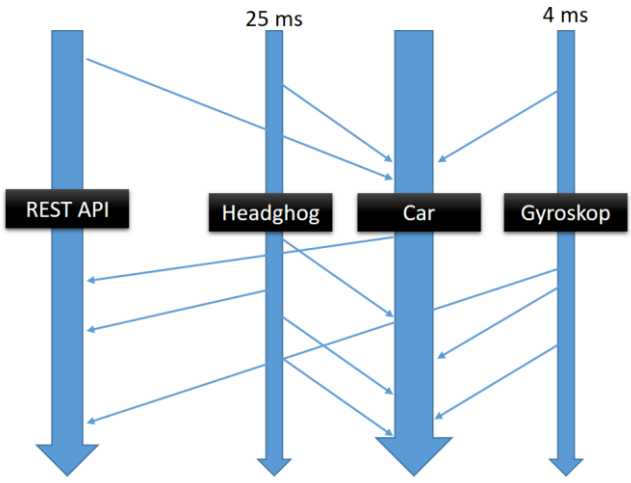
Modul Car se stará o ovládání motorů a jeho součástí je i klíčový algoritmus pro pohyb robota z bodu do bodu. Kromě toho, že disponuje metodami pro nastavení natočení předních kol, otočení celého robota, nastavení rychlosti pohonu, a to vše v relativní a v absolutní podobě, tak disponuje také klíčovou metodou pro nastavení cesty pohybu. Cesta je reprezentována polem souřadnic, přičemž pořadí v poli určuje i pořadí, v jakém se má robot přes tyto body vydat. Jamile je robotovi zadán příkaz pro pohyb přes tyto body, tak se zastaví veškerá činnost robota a robot se okamžitě vydá po dané cestě do pohybu. Konkrétní algoritmus pro pohyb bude popsán níže.

MarvelmindHedge modul je přímo knihovna od firmy Marvelmind pro vyčítání hodnot z mobilního majáku. Knihovna byla pouze pro účely této práce upravena, aby poskytovala lepší funkce rozhraní. Původní součástí knihovny byly pouze dvě funkce, přičemž jedna, veřejná, sloužila pouze pro získání aktuálně uložené polohy majáku a druhá, privátní, sloužila pro aktualizaci proměnné s polohou. Funkce pro aktualizaci polohy se vykonává cyklicky každých 25 ms a stará se o načtení aktuální polohy z mobilního majáku včetně veškeré validace a kontrolních součtů. Pro účely této práce byla pouze upravena funkce pro získání hodnoty z proměnné, kdy se navíc kontroluje, zda je rozdíl mezi posledním časem vyčtení hodnot z majáku a aktuálním časem menší než 30 ms. Pokud je čas větší než 30 ms, je vyvolána výjimka.

Posledním modulem je modul Gyro, který slouží pro vyčítání hodnot z čidla MPU 9250. Hlavní součástí tohoto modulu je opět cyklicky vykonávaná funkce každé 4 ms. V každém cyklu je z gyroskopického čidla vyčtena aktuální rychlost otáčení okolo osy z a z ní je vypočítána relativní změna natočení robota. Tato relativní změna otočení je přičtena (odečtena) k absolutnímu natočení robota. Při spuštění aplikace je absolutní natočení robota nastaveno na 0. Aby modul podával vždy

absolutní orientaci robota vůči souřadnému systému lokalizačního systému, disponuje modul ještě metodou pro korekci orientace. Pokud je této metodě předána hodnota s úhlem od 0 do 360 °, je tento úhel nastaven jako aktuální absolutní orientace. Následně jsou tedy relativní změny naměřené přímo z gyroskopického čidla aplikovány právě na tuto kalibrovanou orientaci. V praxi to tedy funguje tak, že poloha je jednou za čas upravena přímo orientací získanou výpočtem ze změny polohy robota, ale v kratším časovém okamžiku a zejména při otáčení na místě je tato orientace upravována každé 4 ms o relativní změnu z gyroskopického čidla. Kalibrace tedy zaprvé slouží pro sjednocení souřadného systému, ale také jako prevence proti kumulované chybě měření z čidla.

Jak již bylo napsáno výše, každý modul běží v separátním vlákně. Dvě vlákna jsou určena pro vyčítání aktuálních hodnot z čidel, protože hodnoty jsou vyčítány velice často a nepustily by v tomto případě ostatní klíčové prvky logiky vůbec ke zpracování procesorem. Obsluha http serveru s rest API musí běžet také zvlášť ve vlákně, protože musí neustále očekávat dotazy ze sítě a zpracovávat je. Díky tomu, že tato 3 vlákna jsou separovaná, může hlavní vlákno s ovládáním robota reagovat relativně okamžitě na veškeré změny a zadávat okamžité příkazy akčním prvkům. Je potřeba si také uvědomit, že mikropočítač Raspberry disponuje čtyřjádrovým procesorem, tudíž je využití 4 vláknové aplikace optimální. I když je potřeba počítat s tím, že je nepravděpodobné vzhledem k dalším procesům operačního systému, že by běžela všechna 4 vlákna plně paralelně.

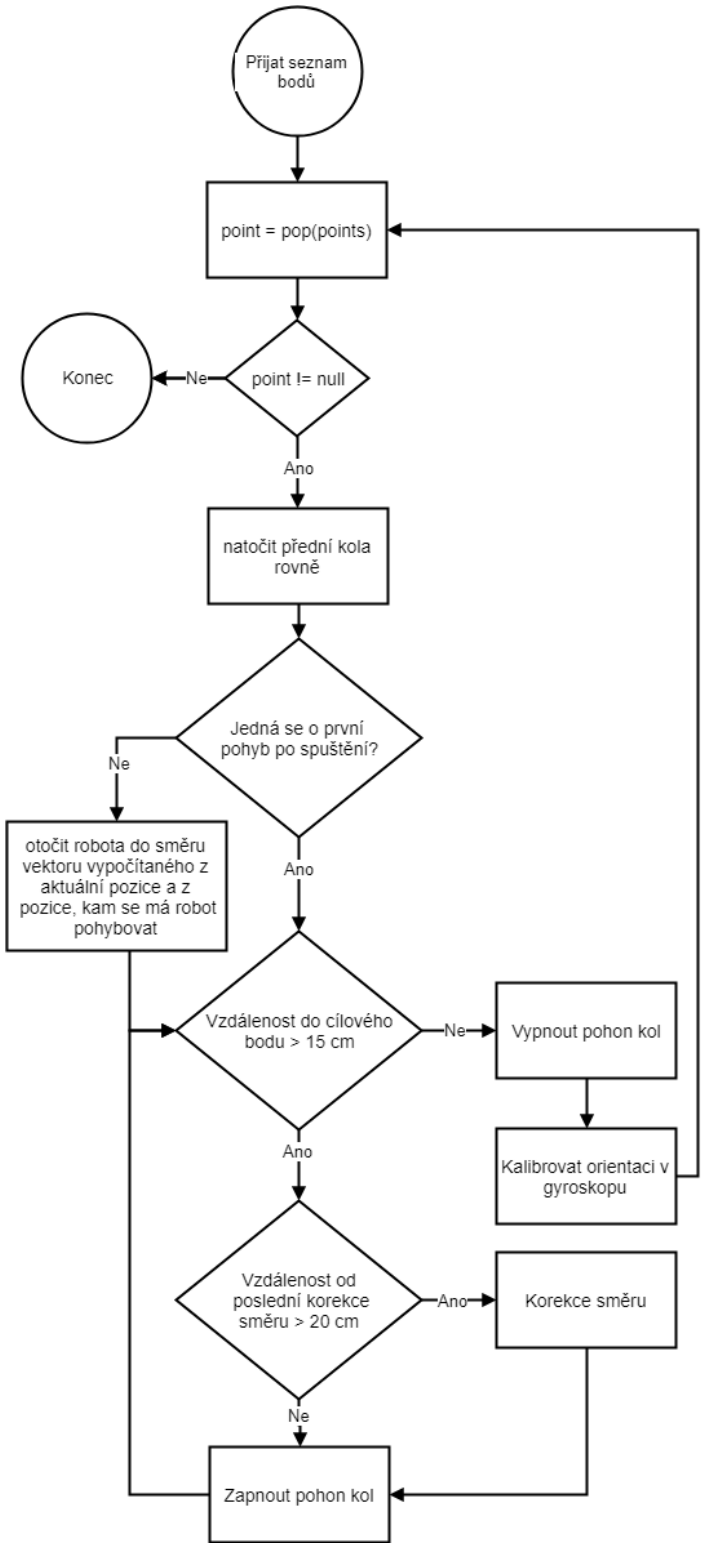


Obrázek 21: Rozdělení logiky robota do vláken

4.3.1 Pohybový algoritmus

Vzhledem k tomu že lokalizační systém Marvelmind podává informaci pouze o aktuální poloze, nikoli o aktuální orientaci, je zapotřebí ještě druhá informace pro detekci orientace. Proto bylo

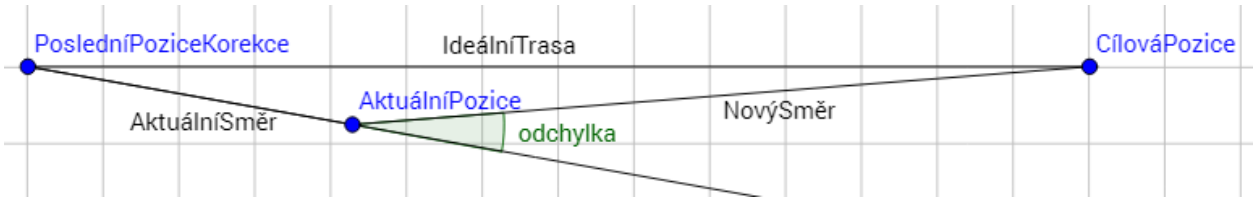
použito čidlo MPU 9250 s gyroskopem, aby mohla být měřena absolutní orientace. Nicméně i gyroskop poskytuje zejména v delším časovém úseku značnou chybovost, protože trpí kumulativní chybou. Z toho důvodu, že jsou obě metody nepřesné, byl navržen algoritmus pro tuto práci, který kombinuje informace z obou čidel.



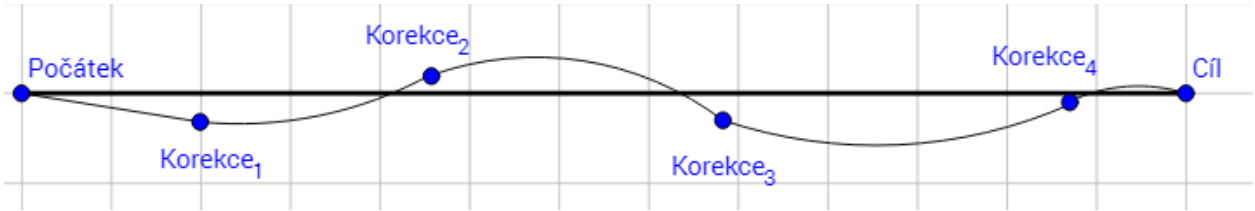
Obrázek 22: Pohybový algoritmus

Algoritmus funguje tak, že gyroskop se využívá pouze pro otáčení robota přímo na místě okolo osy z. Pokud jde o úpravu směru při pohybu, je využíváno výpočtu směru vektoru, který je sestaven z aktuální polohy robota a cílového bodu, přičemž k této korekci dochází každých 20 cm pohybu. Pokud se robot přiblíží k cílovému bodu do vzdálenosti maximálně 15 cm, tak se zastaví, otočí se pomocí informace z gyroskopu do správného směru k dalšímu bodu a pak pokračuje v jízdě. Zároveň při doražení robota do každého bodu je zkalibrována absolutní orientace robota v gyroskopu pomocí skutečné orientace z naposledy naměřeného vektoru. Nevýhoda celého algoritmu spočívá zejména v tom, že pokud se jedná o úplně první pohyb robota po jeho spuštění, nemůže být ještě zkalibrována informace v gyroskopu. Robot se tedy vydá automaticky rovně a teprve až urazí prvních 20 cm, dojde ke kalibraci směru.

Korekce směru, která probíhá každých 20 cm ujeté vzdálenosti, je založená na odchylce aktuálního směru od požadovaného směru. Aktuální směr je vždy spočítán jako vektor z posledního bodu korekce a aktuální polohy. Požadovaný směr je vypočítán jako vektor z aktuální polohy a cílového bodu. Natočení kol je následně nastaveno tak, aby případnou odchylko vyrovnaly. Nicméně vzhledem k relativně dlouhým korekčním intervalům každých 20 cm je již zřejmé, že bude docházet k neustálému regulování natočení kol z jedné strany na druhou. Pokud k tomu připojíme několikacentimetrovou nepřesnost lokalizačního systému, tak je jasné, že robot nemůže tímto způsobem dosáhnout nikdy plynulé rovné jízdy.



Obrázek 23: Odchylka směru



Obrázek 24: Skutečný pohyb robota vůči ideálnímu

5 Cenová náročnost

Předmětem této práce není reálný model robotického zařízení, tudíž není možné určit kompletní cenu. Nicméně cena robotického zařízení se bude odvíjet od jeho konkrétní podoby a jeho podoba může být značně odlišná podle účelu použití. Od samotného provedení robotického modelu se samozřejmě odvíjí i konkrétní typ a model použité baterie a ta tudíž také není součástí cenového návrhu.

Tabulka 2: Cenová náročnost

Prvek	Cena za kus	Potřebný počet kusů
Raspberry Pi 3 Model B	1 046 Kč	1 na robota
MPU 9250	301 Kč	1 na robota
tlačítko	10 Kč	1 na celý systém
Marvelmind router	1 400 Kč	1 na celý systém
Marvelmind mobilní maják	1 400 Kč	1 na robota
Marvelmind statický maják	1 400 Kč	Minimálně 4 podle velikosti haly
Centrální počítač	30 000 Kč	1 na celý systém
Robotické zařízení	???? Kč	1 na robota
Baterie	???? Kč	1 na robota

Z tabulky cen tedy vyplývá, že na jednoho robota je pro implementaci lokalizačního systému zapotřebí 2 747 Kč. Je zapotřebí si také uvědomit, že s Raspberry Pi 3 má tato implementace ještě značné výkonové rezervy a robot může určitě zastat i další individuální funkce. Cena systémových prvků bez prvků na konkrétních robotech je minimálně 37 010 + 1 400 Kč za každé statické čidlo. Odhadem by tedy středně velká hala potřebovala zhruba 30 dalších statických čidel. Celková částka za systém bez robotických zařízení by v tomto případě byla 79 010 Kč.

6 Řešení technických problémů

Jedním z hlavních problémů již na samotném počátku implementace se ukázal být nedostatečný výkon Arduino mikrokontroleru. Zejména bylo znatelné, že je pro něj obrovským zatížením vyřizovat REST API dotazy. Proto byla veškerá REST API komunikace omezena na minimum, aby mikrokontroler zvládal obsluhovat v reálném čase kritické úlohy jako například přímé ovládání motorů.

Jak již bylo řečeno v kapitole Arduino YUN, USB host port je připojen na procesor, a ne přímo na mikrokontroler. Z toho důvodu byl vytvořen skript v jazyce Python, který běží v linuxovém prostředí a čte data z ultrazvukového majáku. Z přijatých dat dekóduje polohu majáku a následně ji přeposílá do mikrokontroleru. Nicméně již v první testovací fázi se ukázalo, že vyčítání dat z USB proudu je velice náročné a výkon procesoru nestačí zpracovávat data v reálném čase (načtení aktuální polohy mělo až dvou sekundové zpoždění). Bylo zjištěno, že nejnáročnější částí celého algoritmu knihovny je výpočet kontrolního součtu. Výpočet kontrolního součtu byl z knihovny odstraněn a zpoždění tím bylo absolutně odstraněno.

Při implementaci dalších periferií na robotickém zařízení a jejich obsluhovacích kódů včetně potřebných knihoven se ukázalo, že Arduino nedisponuje ani dostatečnou programovou pamětí. Nebylo tudíž možné nahrát do zařízení program včetně všech potřebných knihoven. Takže kromě toho, že Arduino zařízení trpělo značnými výkonovými problémy, tak již nebylo možné do něj nahrát obslužný program kvůli nedostatečné velikosti paměti. Arduino umožňuje rozšířit datovou paměť pomocí SD karty, ale neumožňuje rozšířit programovou paměť. Z toho důvodu bylo definitivně od řešení s Arduino deskou upuštěno a bylo nahrazeno výkonnějším zařízením Raspberry Pi 3.

Při testování bylo zjevné, že lokalizační systém Marvelmind funguje relativně dobře a přesně za přívětivých podmínek. Jeho přesnost značně klesá s přibývajícími překážkami a dalšími rušivými jevy v ultrazvukovém spektru. Deklarované přesnosti ± 2 cm lze tedy dosáhnout pouze za ideálních podmínek. Problém je, že v případě rušení nedojde pouze k výpadku signálu, ale většinou spíše k nahlášení chybné polohy. Vzhledem k tomu, že je ze dvou naměřených poloh v pohybovém algoritmu vypočítáván směr pohybu, je přesnost lokalizačního systému kritická a má značný vliv na kvalitu pohybu robota.

Při testovacím měření ve výrobní hale, kde bylo několik rozměrných strojů, které produkovaly při své práci velké množství stlačeného vzduchu, se ukázalo, že zvuk vydávaný stlačeným vzduchem značně ruší ultrazvukový signál. Pro eliminaci rušení byla statická čidla z toho důvodu umístěna

v mnohem menších vzdálenostech, než je doporučováno, zhruba po jednom metru, a tím byl schopný celý systém rušení odolat. Nicméně cenová náročnost takového řešení už značně stoupá.

Kromě lokalizačního systému byl bohužel značný problém i s přesností a chybovostí čidla MPU 9250. Existují sofistikované algoritmy, které kombinují informaci z gyroskopu, akcelerometru a magnetometru a dosahují výborných výsledků. Jeden takový, konkrétně od autora Sebastiana Magwicka, byl implementován na tomto řešení. Nicméně bylo zjištěno, že hodnoty magnetometru jsou silně rušeny elektromotorem, což bylo vyřešeno umístěním čidla dále od motoru. Přesto magnetometr selhával i v blízkostí velkých kovových konstrukcí a v průmyslu by mohl být rušen velkými průmyslovými stroji. Z toho důvodu byl magnetometr vypnut. Magwickův algoritmus nabízí i korekci pouze pomocí gyroskopu a akcelerometru, ale toto řešení také nepodávalo kvalitní výsledky. Z toho důvodu bylo řešení omezeno pouze na data z gyroskopu.

Zásadním problémem však zůstává, že gyroskop podává informace o aktuální rychlosti otáčení, nikoli o aktuální orientaci. Je tedy potřeba dopočítávat aktuální orientaci ze změny rychlosti, k čemuž je však zapotřebí velice často snímat hodnoty z gyroskopu v co nepřesnější délce periody. První požadavek by splnitelný byl, ale přesnost časové periody v systému se standartním operačním systémem a vyšším programovacím jazykem je téměř nemožná. Gyroskop je tedy využíván pouze v případě, kdy je s robotem otáčeno přímo na místě a není tudíž možné dopočítávat orientaci ze změny polohy lokalizačního systému.

Závěr

V úvodní části této práce byly vypracovány rešerše lokalizačních systémů, a to jak pro určení polohy, tak pro určení orientace robota. Jako nejvhodnější kandidát z pohledu cenové náročnosti a flexibility systému byl vybrán lokalizační systém Marvelmind, který funguje na principu měření doby přenosu ultrazvukového signálu. Vzhledem k tomu, že Marvelmind systém dokáže podávat informace pouze o poloze, nikoli o orientaci zařízení, byla vypracována ještě jedna rešerše pro měření orientace robota. Jako nejvhodnější metoda pro měření orientace bylo vybráno měření pomocí gyroskopu.

Navržení kompletní architektury systému včetně komunikačních rozhraní bylo klíčovou částí implementace. Centrální aplikace, která celý systém ovládá, komunikuje s robotem přes standardní WIFI síť a zároveň dostává přímo z Marvelmind systému informace o poloze každého robota. Na robotovi je umístěn mikropočítač, který se stará o veškerou logiku pohybu a ovládání periferií robota.

Po navržení celého konceptu systému bylo zapotřebí vybrat konkrétní hardwarové prostředky. Zatímco RC model pro simulaci robota byl daný již před začátkem práce, ostatní technologie bylo potřeba teprve zvolit. Pro konkrétní realizaci gyroskopického čidla bylo vybráno univerzální čidlo MPU 9250, které zároveň disponuje akcelerometrem a magnetometrem. Největším úskalím celé práce se ukázal být výběr mikropočítače. V první fázi byl pro realizaci mikropočítače vybrán mikropočítač Arduino YUN, který ale trpěl nedostatečným výkonem, zbytečně komplikovanou architekturou a ve výsledku i nedostatečnou programovou pamětí. Proto byl tento mikropočítač nahrazen modelem Raspberry Pi 3, který má jednodušší architekturu a disponuje znatelně vyšším výkonem. Následně bylo navrženo kompletní zapojení v rámci robotického zařízení, kde je většina periferií připojena přímo k mikropočítači. Jedinou výjimkou je motor pro pohon kol, který je připojen přes řídící jednotku RC modelu.

Pro ovládání celého systému byla vytvořena desktopová aplikace s názvem Factory Sheduler v jazyce C#, která zadává robotům příkazy a zároveň umožňuje sledovat jejich stav. Pomocí centrální aplikace se také vytváří virtuální mapa. Dále byly vytvořeny algoritmy pro obsluhu robotického zařízení. Tyto algoritmy jsou napsány vy jazyce Python a jsou spouštěny na mikropočítači.

Zásadním úskalím této práce se stala nepřesnost lokalizačního systému v kombinaci s nepřesností gyroskopického čidla. Lokalizační systém dokáže dostát slibované přesnosti ± 2 cm pouze za ideálních podmínek a s minimem překážek. Gyroskopické čidlo trpí zejména kumulativní chybou měření. Z toho důvodu byl pohybový algoritmus navržen tak, aby se gyroskopické čidlo používalo

pouze pro otáčení robota na místě. Směr robota při pohybu je každých 20 cm jízdy vypočítáván ze změny polohy a tímto vypočítaným směrem je kalibrováno i gyroskopické čidlo, aby se tak odstraňovala kumulativní chyba měření. Nicméně již kvůli zmíněné nepřesnosti lokalizačního systému je jasné, že výpočet směru ze změny polohy nemůže být nikdy zcela přesný. Z toho důvodu není jízda robota po přímce rovná, protože robot neustále koriguje svůj pohyb z leva doprava a dochází tak k jakémusi vlnitému pohybu robota z jednoho bodu do druhého.

Cíl práce se tedy podařilo splnit, i když nasazení v praxi je v současném provedení problematické. Marvelmind lokalizační systém je příliš nespolehlivý a jeho přesnost je značně proměnlivá. Pohyb robota je příliš nerovnoměrný na to, aby se pohyboval v prostorách výrobní haly. Systém by mohl být využitelný v praxi, pokud by byl doplněn vedlejší lokalizační systém, který by umožňoval přesný pohyb robota. Například vodící prvky na podlaze nebo na stropě by mohly být v tomto případě dostačující kompenzací, ale tím by systém ztratil na flexibilitě.

Použitá literatura

1. ARDUINO S.R.L. 2016a. What is Arduino. Arduino [online]. Italy: Arduino S.R.L., [cit. 2016-07-06]. Dostupné z: http://www.arduino.org/learning/getting-started/what-is-arduino
2. ARDUINO S.R.L. 2016b. Arduino YUN. Arduino [online]. Italy: Arduino S.R.L., [cit. 2016-07-06]. Dostupné z: http://www.arduino.org/learning/getting-started/what-is-arduino
3. ARDUINO S.R.L. 2016c. What is a sketch. Arduino [online]. Italy: Arduino S.R.L., [cit. 2016-07-06]. Dostupné z: http://www.arduino.org/learning/getting-started/what-is-arduino
4. ARDUINO S.R.L. 2016d. Getting Started with the Arduino Yún. Arduino [online]. Italy: Arduino S.R.L., [cit. 2016-07-06]. Dostupné z: https://www.arduino.cc/en/Guide/ArduinoYun#toc3
5. AXIAL R/C. 2017. SCX10™ 2012 Jeep® Wrangler Unlimited Rubicon 1/10th Scale Electric. Axial [online]. Los Angeles: Axial R/C, [cit. 2017-05-01]. Dostupné z: http://www.axialracing.com/products/ax90028
6. CHACOS, Brad, 2016. Raspberry Pi 3 review: The revolutionary $35 mini-PC cures its biggest headaches. *PCWorld* [online]. [cit. 2017-04-30]. Dostupné z: http://www.pcworld.com/article/3057888/computers/raspberry-pi-3-review-the-revolutionary-35-mini-pc-cures-its-biggest-headaches.html
7. CNXSoft. 2016. Raspberry Pi 3 Board is Powered by Broadcom BCM2837 Cortex A53 Processor, Sells for $35. *CNXSoft – Embedded Systems News* [online]. [cit. 2017-04-30]. Dostupné z: http://www.cnx-software.com/2016/02/29/raspberry-pi-3-board-is-powered-by-broadcom-bcm2827-cortex-a53-processor-sells-for-35/
8. DIGI-KEY ELECTRONICS. 2014. MPU-9250 9-Axis Gyroscope + Accelerometer + Magnetometer. DigiKey [online]. Digi-Key Electronics [cit. 2017-04-30]. Dostupné z: https://www.digikey.com/en/product-highlight/i/invensense/mpu-9250-9-axis-gyro-accel-magnet
9. KOTARU, Manikanta, Kiran JOSHI, Dinesh BHARADIA a Sachin KATTI. 2015. SpotFi. *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication - SIGCOMM '15* [online]. New York, New York, USA: ACM Press.

269-282 [cit. 2016-07-17]. DOI: 10.1145/2785956.2787487. ISBN 9781450335423. Dostupné z: http://dl.acm.org/citation.cfm?doid=2785956.2787487

10. MALÝ, Martin. 2009. REST: architektura pro webové API. Zdroják.cz [online]. [cit. 2016-07-16]. Dostupné z: https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/

11. MARVELMIND ROBOTICS. 2016. Marvelmind Indoor Navigation System Operating Manual. Sunnyvale. Dostupné také z: http://marvelmind.com/pics/marvelmind_navigation_system_manual.pdf

12. OBERSTEIN, Tobias. 2013. Getting started with Arduino Yun and Autobahn. In: *Tavendo* [online]. [cit. 2016-07-31]. Dostupné z: http://tavendo.com/blog/post/arduino-yun-with-autobahn/

13. OPENSOURCE.COM. 2016. What is a Raspberry Pi? Opensource.com [online]. [cit. 2017-04-30]. Dostupné z: https://opensource.com/resources/what-raspberry-pi

14. RASPBERRY PI FOUNDATION. 2017. GPIO: MODELS A+, B+, RASPBERRY PI 2 B AND RASPBERRY PI 3 B. Raspberry Pi [online]. RASPBERRY PI FOUNDATION, [cit. 2017-04-30]. Dostupné z: https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/

15. SKALKA, Marek. 2011. *Srovnání lokalizačních technik*. Praha. Diplomová práce. Univerzita Karlova v Praze, Matematicko-fyzikální fakulta, Katedra softwarového inženýrství. Vedoucí práce RNDr. David Obdržálek.

16. ŠTĚNO, Jiří. 2012. Využití senzorů v dopravní a manipulační technice. Plzeň. Bakalářská práce. Západočeská univerzita v Plzni. Vedoucí práce Doc. Ing. Jaromír HORÁK, CSc.

Příloha A – Definice REST API pro robota

Tabulka 3: Definice REST API pro robota

URL adresa	HTTP metoda	Popis parametrů	Návratová hodnota	Popis funkce
/check/	GET		"NVC8mK73kAoXzLAYxFMo"	Kontrola připojení
/position/	GET		x,y nebo "NO_HEDGEHOG_CONNECTION_OR_NO_POSITION_UPDATE"	Vrací aktuální pozici robota.
/raid/(-?[0-9]+)	GET	Hodnota od -100 do 100		Natočí přední kola robota relativně o danou hodnotu.
/drive/(-?[0-9]+)	GET	Počet milisekund (záporný = zpátečka)		Pohne s robotem malou rychlostí po daný počet milisekund.
/writeEEPROM /(name|alias|distances)/(.*)	GET	Hodnota parametru. V případě distances se jedná o 4 číselné hodnoty oddělené lomítkem.		Zapíše do persistentní paměti jméno zařízení nebo alias zařízení nebo vzdálenosti okrajů robota od mobilního majáku.
/readEEPROM /(name|alias|distances)	GET		Hodnota parametru. V případě distances se jedná o 4 číselné hodnoty oddělené čárkou.	Přečte z persistentní paměti jméno zařízení nebo alias zařízení nebo

				vzdálenosti okrajů robota od mobilního majáku.
/read-map-points/	GET		NO_SET nebo x,y;x,y;x,y…	Vrátí nastřádané pozice v bufferu.
/turn/(-?[0-9]+)	GET	Počet stupňů.		Otočí robota o daný počet stupňů.
/heading/	GET		0–360 °	Vrátí směr robota ve stupních
/path/	POST	JSON ve formátu: {"path": [{"x": 37, "y": 208}, {"x": 64, "y": 48}]}		Nastaví robotovi body, přes které má jet
/stop/	GET			Zastaví veškerou aktivitu robota.
/max-turn/(-1|1)	GET	Směr rotace		Udělá maximální otočku robota na místě v daném směru.